General sequence data methods

String Methods

In [1]:

```
strSample = 'learning is fun !'
print(strSample)
```

```
learning is fun !
```

In [2]:

```
strSample.capitalize( )  #  returns the string with its first character capitalized and the rest lowercased
```

Out[2]:

```
'Learning is fun !'
```

In [3]:

```
strSample.title()        # to capitalise the first character of each word
```

Out[3]:

```
'Learning Is Fun !'
```

In [4]:

```
strSample.swapcase()     # to swap the case of strings
```

Out[4]:

```
'LEARNING IS FUN !'
```

In [5]:

```
strSample.find('n')      # to find the index of the given letter
```

Out[5]:

```
4
```

In [6]:

```
strSample.count('a')     # to count total number of 'a' in the string
```

Out[6]:

```
1
```

In [7]:

```
strSample.replace('fun','joyful') # to repace the letters/word
```

Out[7]:

```
'learning is joyful !'
```

In [8]:

```
strSample.isalnum()      # Return true if all bytes in the sequence are
                         # alphabetical ASCII characters or ASCII decimal digits,
                         # false otherwise
```

Out[8]:

```
False
```

In [9]:

```
name1  = 'GITAA'
name2  = 'Pvt'
name3  = 'Ltd'
```

In [10]:

```
name='{} {}. {}.'.format(name1,name2,name3)
print(name)
```

```
GITAA Pvt. Ltd.
```

The below code will show all the functions that we can use for the particular variable:

In [11]:

```
print(dir(name))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
'__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__',
'__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul_
_', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitaliz
e', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'fi
nd', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'is
decimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintabl
e', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip',
'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartiti
on', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'sw
apcase', 'title', 'translate', 'upper', 'zfill']
```

In [13]:

```
print(help(str))
```

```
Help on class str in module builtins:

class str(object)
 |  str(object='') -> str
 |  str(bytes_or_buffer[, encoding[, errors]]) -> str
 |
 |  Create a new string object from the given object. If encoding or
 |  errors is specified, then the object must expose a data buffer
 |  that will be decoded using the given encoding and error handler.
 |  Otherwise, returns the result of object.__str__() (if defined)
 |  or repr(object).
 |  encoding defaults to sys.getdefaultencoding().
 |  errors defaults to 'strict'.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __format__(self, format_spec, /)
 |      Return a formatted version of the string as described by format_sp
ec.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(self, key, /)
 |      Return self[key].
 |
 |  __getnewargs__(...)
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __hash__(self, /)
 |      Return hash(self).
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __mod__(self, value, /)
 |      Return self%value.
 |
```

```
 |  __mul__(self, value, /)
 |      Return self*value.
 |
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __rmod__(self, value, /)
 |      Return value%self.
 |
 |  __rmul__(self, value, /)
 |      Return value*self.
 |
 |  __sizeof__(self, /)
 |      Return the size of the string in memory, in bytes.
 |
 |  __str__(self, /)
 |      Return str(self).
 |
 |  capitalize(self, /)
 |      Return a capitalized version of the string.
 |
 |      More specifically, make the first character have upper case and th
e rest lower
 |      case.
 |
 |  casefold(self, /)
 |      Return a version of the string suitable for caseless comparisons.
 |
 |  center(self, width, fillchar=' ', /)
 |      Return a centered string of length width.
 |
 |      Padding is done using the specified fill character (default is a s
pace).
 |
 |  count(...)
 |      S.count(sub[, start[, end]]) -> int
 |
 |      Return the number of non-overlapping occurrences of substring sub
in
 |      string S[start:end].  Optional arguments start and end are
 |      interpreted as in slice notation.
 |
 |  encode(self, /, encoding='utf-8', errors='strict')
 |      Encode the string using the codec registered for encoding.
 |
 |      encoding
 |        The encoding in which to encode the string.
 |      errors
 |        The error handling scheme to use for encoding errors.
 |        The default is 'strict' meaning that encoding errors raise a
 |        UnicodeEncodeError.  Other possible values are 'ignore', 'replac
e' and
 |        'xmlcharrefreplace' as well as any other name registered with
 |        codecs.register_error that can handle UnicodeEncodeErrors.
 |
 |  endswith(...)
 |      S.endswith(suffix[, start[, end]]) -> bool
 |
```

```
 |      Return True if S ends with the specified suffix, False otherwise.
 |      With optional start, test S beginning at that position.
 |      With optional end, stop comparing S at that position.
 |      suffix can also be a tuple of strings to try.
 |
 |  expandtabs(self, /, tabsize=8)
 |      Return a copy where all tab characters are expanded using spaces.
 |
 |      If tabsize is not given, a tab size of 8 characters is assumed.
 |
 |  find(...)
 |      S.find(sub[, start[, end]]) -> int
 |
 |      Return the lowest index in S where substring sub is found,
 |      such that sub is contained within S[start:end].  Optional
 |      arguments start and end are interpreted as in slice notation.
 |
 |      Return -1 on failure.
 |
 |  format(...)
 |      S.format(*args, **kwargs) -> str
 |
 |      Return a formatted version of S, using substitutions from args and
kwargs.
 |      The substitutions are identified by braces ('{' and '}').
 |
 |  format_map(...)
 |      S.format_map(mapping) -> str
 |
 |      Return a formatted version of S, using substitutions from mapping.
 |      The substitutions are identified by braces ('{' and '}').
 |
 |  index(...)
 |      S.index(sub[, start[, end]]) -> int
 |
 |      Return the lowest index in S where substring sub is found,
 |      such that sub is contained within S[start:end].  Optional
 |      arguments start and end are interpreted as in slice notation.
 |
 |      Raises ValueError when the substring is not found.
 |
 |  isalnum(self, /)
 |      Return True if the string is an alpha-numeric string, False otherw
ise.
 |
 |      A string is alpha-numeric if all characters in the string are alph
a-numeric and
 |      there is at least one character in the string.
 |
 |  isalpha(self, /)
 |      Return True if the string is an alphabetic string, False otherwis
e.
 |
 |      A string is alphabetic if all characters in the string are alphabe
tic and there
 |      is at least one character in the string.
 |
 |  isascii(self, /)
 |      Return True if all characters in the string are ASCII, False other
wise.
 |
```

```
 |      ASCII characters have code points in the range U+0000-U+007F.
 |      Empty string is ASCII too.
 |
 |  isdecimal(self, /)
 |      Return True if the string is a decimal string, False otherwise.
 |
 |      A string is a decimal string if all characters in the string are d
ecimal and
 |      there is at least one character in the string.
 |
 |  isdigit(self, /)
 |      Return True if the string is a digit string, False otherwise.
 |
 |      A string is a digit string if all characters in the string are dig
its and there
 |      is at least one character in the string.
 |
 |  isidentifier(self, /)
 |      Return True if the string is a valid Python identifier, False othe
rwise.
 |
 |      Use keyword.iskeyword() to test for reserved identifiers such as
"def" and
 |      "class".
 |
 |  islower(self, /)
 |      Return True if the string is a lowercase string, False otherwise.
 |
 |      A string is lowercase if all cased characters in the string are lo
wercase and
 |      there is at least one cased character in the string.
 |
 |  isnumeric(self, /)
 |      Return True if the string is a numeric string, False otherwise.
 |
 |      A string is numeric if all characters in the string are numeric an
d there is at
 |      least one character in the string.
 |
 |  isprintable(self, /)
 |      Return True if the string is printable, False otherwise.
 |
 |      A string is printable if all of its characters are considered prin
table in
 |      repr() or if it is empty.
 |
 |  isspace(self, /)
 |      Return True if the string is a whitespace string, False otherwise.
 |
 |      A string is whitespace if all characters in the string are whitesp
ace and there
 |      is at least one character in the string.
 |
 |  istitle(self, /)
 |      Return True if the string is a title-cased string, False otherwis
e.
 |
 |      In a title-cased string, upper- and title-case characters may only
 |      follow uncased characters and lowercase characters only cased one
s.
 |
```

```
 |  isupper(self, /)
 |      Return True if the string is an uppercase string, False otherwise.
 |
 |      A string is uppercase if all cased characters in the string are up
percase and
 |      there is at least one cased character in the string.
 |
 |  join(self, iterable, /)
 |      Concatenate any number of strings.
 |
 |      The string whose method is called is inserted in between each give
n string.
 |      The result is returned as a new string.
 |
 |      Example: '.'.join(['ab', 'pq', 'rs']) -> 'ab.pq.rs'
 |
 |  ljust(self, width, fillchar=' ', /)
 |      Return a left-justified string of length width.
 |
 |      Padding is done using the specified fill character (default is a s
pace).
 |
 |  lower(self, /)
 |      Return a copy of the string converted to lowercase.
 |
 |  lstrip(self, chars=None, /)
 |      Return a copy of the string with leading whitespace removed.
 |
 |      If chars is given and not None, remove characters in chars instea
d.
 |
 |  partition(self, sep, /)
 |      Partition the string into three parts using the given separator.
 |
 |      This will search for the separator in the string.  If the separato
r is found,
 |      returns a 3-tuple containing the part before the separator, the se
parator
 |      itself, and the part after it.
 |
 |      If the separator is not found, returns a 3-tuple containing the or
iginal string
 |      and two empty strings.
 |
 |  replace(self, old, new, count=-1, /)
 |      Return a copy with all occurrences of substring old replaced by ne
w.
 |
 |        count
 |          Maximum number of occurrences to replace.
 |          -1 (the default value) means replace all occurrences.
 |
 |      If the optional argument count is given, only the first count occu
rrences are
 |      replaced.
 |
 |  rfind(...)
 |      S.rfind(sub[, start[, end]]) -> int
 |
 |      Return the highest index in S where substring sub is found,
 |      such that sub is contained within S[start:end].  Optional
```

```
 |      arguments start and end are interpreted as in slice notation.
 |
 |      Return -1 on failure.
 |
 |  rindex(...)
 |      S.rindex(sub[, start[, end]]) -> int
 |
 |      Return the highest index in S where substring sub is found,
 |      such that sub is contained within S[start:end].  Optional
 |      arguments start and end are interpreted as in slice notation.
 |
 |      Raises ValueError when the substring is not found.
 |
 |  rjust(self, width, fillchar=' ', /)
 |      Return a right-justified string of length width.
 |
 |      Padding is done using the specified fill character (default is a s
pace).
 |
 |  rpartition(self, sep, /)
 |      Partition the string into three parts using the given separator.
 |
 |      This will search for the separator in the string, starting at the
end. If
 |      the separator is found, returns a 3-tuple containing the part befo
re the
 |      separator, the separator itself, and the part after it.
 |
 |      If the separator is not found, returns a 3-tuple containing two em
pty strings
 |      and the original string.
 |
 |  rsplit(self, /, sep=None, maxsplit=-1)
 |      Return a list of the words in the string, using sep as the delimit
er string.
 |
 |        sep
 |          The delimiter according which to split the string.
 |          None (the default value) means split according to any whitespa
ce,
 |          and discard empty strings from the result.
 |        maxsplit
 |          Maximum number of splits to do.
 |          -1 (the default value) means no limit.
 |
 |      Splits are done starting at the end of the string and working to t
he front.
 |
 |  rstrip(self, chars=None, /)
 |      Return a copy of the string with trailing whitespace removed.
 |
 |      If chars is given and not None, remove characters in chars instea
d.
 |
 |  split(self, /, sep=None, maxsplit=-1)
 |      Return a list of the words in the string, using sep as the delimit
er string.
 |
 |      sep
 |        The delimiter according which to split the string.
 |        None (the default value) means split according to any whitespac
```

```
e,
 |        and discard empty strings from the result.
 |      maxsplit
 |        Maximum number of splits to do.
 |        -1 (the default value) means no limit.
 |
 |  splitlines(self, /, keepends=False)
 |      Return a list of the lines in the string, breaking at line boundar
ies.
 |
 |      Line breaks are not included in the resulting list unless keepends
is given and
 |      true.
 |
 |  startswith(...)
 |      S.startswith(prefix[, start[, end]]) -> bool
 |
 |      Return True if S starts with the specified prefix, False otherwis
e.
 |      With optional start, test S beginning at that position.
 |      With optional end, stop comparing S at that position.
 |      prefix can also be a tuple of strings to try.
 |
 |  strip(self, chars=None, /)
 |      Return a copy of the string with leading and trailing whitespace r
emove.
 |
 |      If chars is given and not None, remove characters in chars instea
d.
 |
 |  swapcase(self, /)
 |      Convert uppercase characters to lowercase and lowercase characters
to uppercase.
 |
 |  title(self, /)
 |      Return a version of the string where each word is titlecased.
 |
 |      More specifically, words start with uppercased characters and all
remaining
 |      cased characters have lower case.
 |
 |  translate(self, table, /)
 |      Replace each character in the string using the given translation t
able.
 |
 |        table
 |          Translation table, which must be a mapping of Unicode ordinals
to
 |          Unicode ordinals, strings, or None.
 |
 |      The table must implement lookup/indexing via __getitem__, for inst
ance a
 |      dictionary or list.  If this operation raises LookupError, the cha
racter is
 |      left untouched.  Characters mapped to None are deleted.
 |
 |  upper(self, /)
 |      Return a copy of the string converted to uppercase.
 |
 |  zfill(self, width, /)
 |      Pad a numeric string with zeros on the left, to fill a field of th
```

```
e given width.
 |
 |      The string is never truncated.
 |
 |  ------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signa
ture.
 |
 |  maketrans(x, y=None, z=None, /)
 |      Return a translation table usable for str.translate().
 |
 |      If there is only one argument, it must be a dictionary mapping Uni
code
 |      ordinals (integers) or characters to Unicode ordinals, strings or
None.
 |      Character keys will be then converted to ordinals.
 |      If there are two arguments, they must be strings of equal length,
and
 |      in the resulting dictionary, each character in x will be mapped to
the
 |      character at the same position in y. If there is a third argument,
it
 |      must be a string, whose characters will be mapped to None in the r
esult.

None
```

In [12]:

```
print(help(str.find))
```

```
Help on method_descriptor:

find(...)
    S.find(sub[, start[, end]]) -> int

    Return the lowest index in S where substring sub is found,
    such that sub is contained within S[start:end].  Optional
    arguments start and end are interpreted as in slice notation.

    Return -1 on failure.

None
```

Sequence datatype object initializations

In [13]:

```
strSample = 'learning is fun !'          # STRING
```

In [14]:

```
lstSample = [1,2,'a','sam',2]            # list
```

In [15]:

```
from array import *
```

In [16]:

```
arrSample = array('i',[1,2,3,4])        # array
```

In [17]:

```
tupSample = (1,2,3,4,3,'py')            # tuple
```

In [18]:

```
dictSample= {1:'first', 'second':2, 3:3, 'four':'4'} # dictionary
```

In [19]:

```
setSample = {'example',24,87.5,'data',24,'data'}      # set

keys      = (1,'second',3,'four')

rangeSample = range(1,12,4)             # built-in sequence type used for looping

for x in rangeSample: print(x)
```

```
1
5
9
```

len(object) returns number of elements in the object

accepted object types: string, list, array, tuple, dictionary, set, range

In [20]:

```
print("No. of elements in the object :")
print("string = {} , list= {}, array= {}, tuple = {},\
      dictionary ={}, set ={}, range ={}".format(len(strSample), len(lstSample),
      len(arrSample) , len(tupSample), len(dictSample), len(setSample), len(rangeSample
)))
```

```
No. of elements in the object :
string = 17 , list= 5, array= 4, tuple = 6,      dictionary =4, set =4, ra
nge =3
```

In [21]:

```
print(lstSample)
```

```
[1, 2, 'a', 'sam', 2]
```

In [23]:

```
lstSample.reverse()        # Reverses the order of the list
print(lstSample)
```

```
[2, 'sam', 'a', 2, 1]
```

The clear() method removes all items from the object

Supported sequence data: list, dictionary, set

In [24]:

```
lstSample.clear()
print(lstSample)
```

```
[]
```

In [25]:

```
dictSample.clear()
print(dictSample)
```

```
{}
```

In [26]:

```
setSample.clear()
print(setSample)
```

```
Traceback (most recent call last):

  File "<ipython-input-26-60ed937b48b7>", line 1, in <module>
    setSample.clear()

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

append() Adds an element at the end of the object

Supported datatypes: array, list, set

In [27]:

```
arrSample.append(3)            # adding an element, 3 to the 'arrSample'
print(arrSample)               # updated array, arrSample
```

```
array('i', [1, 2, 3, 4, 3])
```

In [28]:

```
print(lstSample)
```

```
[]
```

In [29]:

```
lstSample.append([2,4])        # adding [2, 4] list to lstSample
print(lstSample)               # updated list
```

```
[[2, 4]]
```

In [34]:

```
#setSample.append(20)           # AttributeError: 'set' object has no attribute 'appen
d'
setSample.add(20)              # add() takes single parameter(element) which needs to
                               # be added in the set
print(setSample)
```

```
{'data', 20, 87.5, 24, 'example'}
```

update() function in set adds elements from a set (passed as an argument) to the set

- This method takes only single argument
- The single argument can be a set, list, tuples or a dictionary
- It automatically converts into a set and adds to the set

In [35]:

```
setSample.update([5,10])       # adding a list of elements to the set
print(setSample)               # updated set
```

```
{'data', 5, 10, 20, 87.5, 24, 'example'}
```

Dictionary Methods

In [22]:

```
print(dictSample)
```

```
{1: 'first', 'second': 2, 3: 3, 'four': '4'}
```

In [23]:

```
dictSample["five"]  = 5
print(dictSample)
```

```
{1: 'first', 'second': 2, 3: 3, 'four': '4', 'five': 5}
```

In [24]:

```
dictSample.update(five = 5)    # update the dictionary with the key/value pairs from oth
er,
                               # overwriting existing keys
print(dictSample)              # updated dictionary
```

```
{1: 'first', 'second': 2, 3: 3, 'four': '4', 'five': 5}
```

In [25]:

```
list(dictSample)              # returns a list of all the keys used in the dictionary d
ictSample
```

Out[25]:

```
[1, 'second', 3, 'four', 'five']
```

In [26]:

```
len(dictSample)               # returns the number of items in the dictionary
```

Out[26]:

```
5
```

In [27]:

```
dictSample.get("five")        # it is a conventional method to access a value for a key
```

Out[27]:

```
5
```

In [28]:

```
dictSample.keys()             # returns list of keys in dictionary
```

Out[28]:

```
dict_keys([1, 'second', 3, 'four', 'five'])
```

In [ ]:

```
dictSample.items()            # returns a list of (key, value) tuple pairs
```

insert()- inserts the element at the specified index of the object

- supported datatypes: array, list

In [29]:

```
print(arrSample)
```

```
array('i', [1, 2, 3, 4])
```

In [30]:

```
arrSample.insert(1,100)       # inserting the element 100 at 2nd position
print(arrSample)              # printing array
```

```
array('i', [1, 100, 2, 3, 4])
```

In [31]:

```
lstSample.insert(5,24)        # inserting the element 24 at 5th position
print(lstSample)              # printing list
```

```
[1, 2, 'a', 'sam', 2, 24]
```

pop()- removes the element at the given index from the object and prints the same

- default value is -1, which returns the last item
- supported datatypes: array, list, set, dictionary

In [32]:

```
arrSample.pop()               # deleting the last element and prints the same
```

Out[32]:

```
4
```

In [33]:

```
print(lstSample)
lstSample.pop(4)              # deleting the 5th element from the list
```

```
[1, 2, 'a', 'sam', 2, 24]
```

Out[33]:

```
2
```

In [34]:

```
print(dictSample)
dictSample.pop('second')      # deleting the key - second
```

```
{1: 'first', 'second': 2, 3: 3, 'four': '4', 'five': 5}
```

Out[34]:

```
2
```

In [35]:

```
dictSample.pop(3)             # deleting the key - 3
```

Out[35]:

```
3
```

Set is an unordered sequence and hence pop is not usually used

The remove() method removes the first occurrence of the element with the specified value

- supported datatypes: array, list, dictionary, set

In [36]:

```
print(arrSample)
```

```
array('i', [1, 100, 2, 3])
```

In [37]:

```
arrSample.remove(0)             # ValueError: array.remove(x): x not in list
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-37-1320e2f283e2> in <module>
----> 1 arrSample.remove(0)             # ValueError: array.remove(x): x n
ot in list

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

In [38]:

```
arrSample.remove(2)             # removes the element 2 from the array, arrSample
print(arrSample)
```

```
array('i', [1, 100, 3])
```

In [39]:

```
print(lstSample)
lstSample.remove('sam')         # removes the element 'sam' from the list, lstSample
print(lstSample)
```

```
[1, 2, 'a', 'sam', 24]
[1, 2, 'a', 24]
```

In [40]:

```
print(setSample)
setSample.remove(57)            # KeyError: 57
```

```
{'example', 24, 'data', 87.5}

---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
<ipython-input-40-46fc2eeaf51b> in <module>
      1 print(setSample)
----> 2 setSample.remove(57)            # KeyError: 57

KeyError: 57
```

In [41]:

```
setSample.discard(57)           # The set remains unchanged if the element passed to
                                # discard() method doesn't exist
print(setSample)
```

```
{'example', 24, 'data', 87.5}
```

del: deletes the entire object of any data type

- syntax: del obj_name
- del is a Python keyword
- obj_name can be variables, user-defined objects, lists, items within lists, dictionaries etc.

In [42]:

```
del setSample                   # deleting the set, setSample
print(setSample)                # NameError: name 'setSample' is not defined
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-42-a69daeb3bd1f> in <module>
      1 del setSample                   # deleting the set, setSample
----> 2 print(setSample)                # NameError: name 'setSample' is n
ot defined

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

In [43]:

```
del arrSample                    # deleting the array, arrSample
print(arrSample)                 # NameError: name 'arrSample' is not defined
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-43-259a5a28b0c1> in <module>
      1 del arrSample                    # deleting the array, arrSample
----> 2 print(arrSample)                 # NameError: name 'arrSample' is n
ot defined

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

In [44]:

```
del lstSample                    # deleting the list, lstSample
print(lstSample)                 # NameError: name 'lstSample' is not defined
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-44-0e560d24951c> in <module>
      1 del lstSample                    # deleting the list, lstSample
----> 2 print(lstSample)                 # NameError: name 'lstSample' is n
ot defined

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

In [45]:

```
del lstSample[2]                 # deleting the third item
print(lstSample)
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-45-c5c817bf9009> in <module>
----> 1 del lstSample[2]                 # deleting the third item
      2 print(lstSample)

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

In [46]:

```
del lstSample[1:3]              # deleting elements from 2nd to 4th
print(lstSample)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-46-3f7b8caa7840> in <module>
----> 1 del lstSample[1:3]              # deleting elements from 2nd to 4th
      2 print(lstSample)

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

In [47]:

```
del lstSample[:]                # deleting all elements from the list
print(lstSample)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-47-f55596c6466c> in <module>
----> 1 del lstSample[:]                # deleting all elements from the list
      2 print(lstSample)

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

In [48]:

```
del dictSample                  # deleting the dictionary, dictSample
print(dictSample)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-48-fe4722cf2eca> in <module>
      1 del dictSample                  # deleting the dictionary, dictSample
----> 2 print(dictSample)

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

The extend() method adds the specified list elements (or any iterable - list, set, tuple, etc.) to the end of the current list

Array operations

In [10]:

```
arrSample.extend((4,5,3,5))     # add a tuple to the arrSample array:
print(arrSample)
```

```
array('i', [1, 2, 3, 4, 4, 5, 3, 5, 4, 5, 3, 5])
```

In [11]:

```
print(arrSample.extend(['sam']))  # TypeError: an integer is required (got type str) -
 should match with itemcode of the array
```

```
Traceback (most recent call last):

  File "<ipython-input-11-9f9b870f6629>", line 1, in <module>
    print(arrSample.extend(['sam']))  # TypeError: an integer is required
 (got type str) - should match with itemcode of the array

TypeError: an integer is required (got type str)
```

In [13]:

```
arrSample.fromlist([3, 4])      # add values from a list to an array
print(arrSample)

arrSample.tolist()              # to convert an array into an ordinary list with the sa
me items
```

```
array('i', [1, 2, 3, 4, 4, 5, 3, 5, 4, 5, 3, 5, 3, 4, 3, 4])
```

Out[13]:

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

In [ ]:

```
range
```

List operations

Create a list using loop and function

In [15]:

```
def numbers(n):
    return [i for i in range(1, n+1)]
```

In [16]:

```
numbers(10)
```

Out[16]:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Set operations

setSample

- A set is an unordered collection of items
  - Every element is unique (no duplicates)
  - Sets can be used to perform mathematical set operations like union, intersection, symmetric difference etc

In [17]:

```
A = {'example',24,87.5,'data',24,'data'} # set of mixed data types
print(A)
```

```
{24, 'data', 'example', 87.5}
```

In [18]:

```
B = {24, 87.5}            # set of integers
print(B)
```

```
{24, 87.5}
```

In [20]:

```
print(A | B)              # union of A and B is a set of all elements from both sets
A.union(B)                # using union() on B
```

```
{'example', 87.5, 24, 'data'}
```

Out[20]:

```
{24, 87.5, 'data', 'example'}
```

In [21]:

```
print(A & B)              # intersection of A and B is a set of elements that are commo
n in both sets
A.intersection(B)         # using intersection() on B
```

```
{24, 87.5}
```

Out[21]:

```
{24, 87.5}
```

END OF SCRIPT