String Datatype :- It is an immutable sequence of characters, where space is also a character.

Each character has a specific position number called index, and the index is used to address the character.
The index value starts from 0.

A string can be assigned using ' ', " ", ''' ''', or """ """.

In [1]:
```
s = 'balloon is red'
type(s)
```

Out[1]: str

In [2]:
```
s1 = "A"
type(s1)
```

Out[2]: str

In [4]:
```
# for multi-line strings:-
z = """this
is
a multi-line
string."""
type(z)
```

Out[4]: str

String Indexing :- a process of accessing any specific element using its index position.

Syntax:
varname[index_position]

In [5]:
```
z = 'AEROPLANE'
z[4]
```

Out[5]: 'P'

In [6]:
```
z = 'AEROPLANE'
z[-1]
```

Out[6]: 'E'

String Slicing: taking out a subpart from any whole.

syntax : var[start_index : stop_index+1 : step]

In [7]:
```
z = 'AEROPLANE'
z[4:9:1]
```

Out[7]: 'PLANE'

In [9]:
```
z = 'AEROPLANE'
# 'ARPAE'
z[0:9:2]
```

Out[9]: 'ARPAE'

In [11]:
```
z = 'AEROPLANE'
# 'ENALPOREA'
z[-1:-10:-1]
```

Out[11]: 'ENALPOREA'

In [14]:
```
# the default value of start index is the starting index
# of the string in any direction.
z = 'AEROPLANE'
print(z[:4:1])
print(z[:-4:-1])
```

```
AERO
ENA
```

In [17]:
```
# the default value of the stop index is the ending character
# of the string in any direction.
z = 'AEROPLANE'
print(z[0::2])
print(z[-1::-2])
```

```
ARPAE
EAPRA
```

In [21]:
```
#the default value of step is 1 in +ve direction.
z = 'AEROPLANE'
print(z[1:9])
print(z[-1:-10])
```

```
EROPLANE
```

Q. Find the follwing substring from the string :

'welcome to my blog

1. 'loet'
2. 'om lg'
3. 'golb ym ot emoclew'

String Attributes:-

In [2]:
```
# upper(): converts all characters to uppercase.
z = 'Sky is Blue'
z.upper()
```

Out[2]: 'SKY IS BLUE'

In [3]:
```
print(z)
```

```
Sky is Blue
```

In [5]:
```
# lower(): converts all characters to lowercase.
z = 'Sky is Blue'
z.lower()
```

Out[5]: 'sky is blue'

In [7]:
```
# title(): convert each word's first character to uppercase.
z = 'Sky is Blue'
z.title()
```

Out[7]: 'Sky Is Blue'

In [8]:
```
# capitalize(): convert only the first character of the string to uppercase.
z = 'Sky is Blue'
z.capitalize()
```

Out[8]: 'Sky is blue'

In [10]:
```
# swapcase(): converts the lowercase to uppercase & vice-versa
z = 'Sky is Blue'
z.swapcase()
```

Out[10]: 'sKY IS bLUE'

In [15]:
```
# count(): it returns the no.of repeatation of any specific element.
z = 'Skyyyyy is Blue'
z.count('Blue')
```

Out[15]: 1

In [17]:
```
# index(): it returns the index poistion of given element.
z = 'Sky is Blue'
z.index('B')
```

Out[17]: 7

In [18]:
```
z.index('x')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[18], line 1
----> 1 z.index('x')

ValueError: substring not found
```

In [20]:
```
# strip(): removes all the unwanted spaces from the leading and
# trailing part of the string.
x = '       pink        '
x.strip()
```

Out[20]: 'pink'

In [23]:
```
# lstrip(): removes all the unwanted spaces from the leading
# part of the string.
x = '       pink        '
x.lstrip()
```

Out[23]: 'pink   '

In [24]:
```
# rstrip(): removes all the unwanted spaces from
# trailing part of the string.
x = '       pink        '
x.rstrip()
```

Out[24]: '   pink'

In [25]:
```
x1 = "**********%%%*****hello***%%%%%**************"
x1.strip('*%')
```

Out[25]: 'hello'

In [26]:
```
z = 'ice'
z.center(50,'*')
```

Out[26]: '**********************ice**********************'

In [27]:
```
z = 'ice'
z.center(50,'🐔')
```

Out[27]: '🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔ice🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔🐔'

In [28]:
```
# join:
x = ['lily','rose','lavendar','orchid']
' * '.join(x)
```

Out[28]: 'lily * rose * lavendar * orchid'

In [30]:
```
# split(): be default separates the words from space.
'lily rose jasmine'.split()
```

Out[30]: ['lily', 'rose', 'jasmine']

In [32]:
```
'lily,rose,jasmine'.split(',')
```

Out[32]:
```
['lily', 'rose', 'jasmine']
```

In [33]:
```
# isupper():-
'BALL'.isupper()
```

Out[33]:
```
True
```

In [34]:
```
'rose'.isupper()
```

Out[34]:
```
False
```

In [35]:
```
# islower():
'tiger'.islower()
```

Out[35]:
```
True
```

In [36]:
```
'Hello'.islower()
```

Out[36]:
```
False
```

In [37]:
```
# istitle():
'The Sky'.istitle()
```

Out[37]:
```
True
```

In [38]:
```
# isalpha():
'ABCjghsk'.isalpha()
```

Out[38]:
```
True
```

In [39]:
```
'abc 677'.isalpha()
```

Out[39]:
```
False
```

In [40]:
```
# isnumeric():
'65767877'.isnumeric()
```

Out[40]:
```
True
```

In [41]:
```
'6767878 7898'.isnumeric()
```

Out[41]:
```
False
```

In [42]:
```
# isdigit():
'67678787898'.isdigit()
```

Out[42]:
```
True
```

In [43]:
```
# isalnum():
'alpha567889'.isalnum()
```

Out[43]: True

In [44]:
```
'alpha % 567889'.isalnum()
```

Out[44]: False

In [46]:
```
# isidentifier():-
'var@123'.isidentifier()
```

Out[46]: False

In [47]:
```
'var_1278'.isidentifier()
```

Out[47]: True

List Datatype :- It is a sequence of items, inside a [ ], items are separated by comma.

A list is mutable : items can be modified.
A list is indexed : items have index numbers that can be used for indexing and slicing.

In [48]:
```
x = [56,90.99,'string','monkey',90+88j]
type(x)
```

Out[48]: list

In [49]:
```
# indexing:-
x = [56,90.99,'string','monkey',90+88j]
x[3]
```

Out[49]: 'monkey'

In [50]:
```
x[-4]
```

Out[50]: 90.99

In [51]:
```
# Slicing:-
# varname[start_index : stop_index+1 : step]
x = [56,90.99,'string','monkey',90+88j]
x[::-1]
```

Out[51]: [(90+88j), 'monkey', 'string', 90.99, 56]

In [52]:
```
x[::]
```

Out[52]: [56, 90.99, 'string', 'monkey', (90+88j)]

List Attributes :-

Adding items to list :

```
append(ob)           Adds an element at the end of the list
extend(seq)        Add the elements of a list (or any iterable), to
the end of the current list
insert(index,ob)   Adds an element at the specified position
```

Removing items from list :

```
clear()    Removes all the elements from the list
pop(index)      Removes the element at the specified position
remove(ob)    Removes the first item with the specified value
```

Miscellaneous functions :

```
copy()     Returns a copy of the list
count(ob)       Returns the number of elements with the specified
value
index(ob)       Returns the index of the first element with the
specified value
reverse()    Reverses the order of the list
sort()     Sorts the list in increasing order
sort(reverse= True) Sort the list in decreasing order
```

In [55]:
```
# append(new_element): it add to the last of the list.
x = [56,90.99,'string','monkey',90+88j]
x.append('dog')
x.append(6789)
print(x)
```

```
[56, 90.99, 'string', 'monkey', (90+88j), 'dog', 6789]
```

In [56]:
```
# insert(index+position,new_element):
x = [56,90.99,'string','monkey',90+88j]
x.insert(1,'ice')
print(x)
```

```
[56, 'ice', 90.99, 'string', 'monkey', (90+88j)]
```

In [57]:
```
# extend([list of elements]):
x = [56,90.99,'string','monkey',90+88j]
x.extend(['a','b','c'])
print(x)
```

```
[56, 90.99, 'string', 'monkey', (90+88j), 'a', 'b', 'c']
```

In [59]:
```
# Deleting of items:
# remove(element):
x = [56,90.99,'string','monkey',90+88j,90.99]
x.remove(90.99)
x
```

Out[59]: [56, 'string', 'monkey', (90+88j), 90.99]

In [60]:
```
# pop(index_position):
x = [56,90.99,'string','monkey',90+88j,90.99]
x.pop(-1)
```

Out[60]: 90.99

In [61]:
```
# clear():
x = [56,90.99,'string','monkey',90+88j,90.99]
x.clear()
x
```

Out[61]: []

In [62]:
```
# count():
x = [56,90.99,'string','monkey',90+88j,90.99]
x.count(90.99)
```

Out[62]: 2

In [63]:
```
# reverse():
x = [56,90.99,'string','monkey',90+88j,90.99]
x.reverse()
x
```

Out[63]: [90.99, (90+88j), 'monkey', 'string', 90.99, 56]

In [65]:
```
# sort(): arrange in ascending order.
x = [56,90,78,456,909,356,234]
x.sort()
x
```

Out[65]: [56, 78, 90, 234, 356, 456, 909]

In [66]:
```
x = [56,90,78,456,909,356,234]
x.sort(reverse=True)
x
```

Out[66]: [909, 456, 356, 234, 90, 78, 56]

In [67]:
```
x = [56,90,78,456,909,356,234]
x.sort(reverse=99)
x
```

Out[67]: [909, 456, 356, 234, 90, 78, 56]

In [68]:
```
x = [56,90,78,456,909,356,234]
x.sort(reverse=0)
x
```

Out[68]: [56, 78, 90, 234, 356, 456, 909]

Tuple :- It is an immutable ordered sequence of items in a ().

1. Items are indexed.
2. Items can't be changed.

In [69]:
```
tup = (12,89,'snow','ice',89.7j)
type(tup)
```

Out[69]: tuple

In [70]:
```
# Indexing:-
tup = (12,89,'snow','ice',89.7j)
tup[-3]
```

Out[70]: 'snow'

In [71]:
```
tup[2]
```

Out[71]: 'snow'

In [73]:
```
# Slicing:
# Syntax: varname[start_index : stop_index+1 : step]
tup = (12,89,'snow','ice',89.7j)
tup[::-1]
```

Out[73]: (89.7j, 'ice', 'snow', 89, 12)

In [74]:
```
tup[::2]
```

Out[74]: (12, 'snow', 89.7j)

Tuple Attributes :

count() The number of times the specified element occurs in the tuple
index() Returns index place of the first occurance of item

In [75]:
```
# count():
tup = (12,89,'snow','ice',89.7j)
tup.count('ice')
```

Out[75]: 1

In [77]:
```
# index():
tup = (12,89,'snow','ice',89+7j,89)
tup.index(89)
```

Out[77]: 1

In [78]:
```
tup = (12,89,'snow','ice',89.7j)
tup.reverse()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[78], line 2
      1 tup = (12,89,'snow','ice',89.7j)
----> 2 tup.reverse()

AttributeError: 'tuple' object has no attribute 'reverse'
```

Set :- It is a collection of items inside a {}.

1. Set items are unordered.
2. Items are unchangable.
3. Do not allow duplicate values.
4. Supports all mathematical set operation.

In [79]:
```
s = {12,89,'snow','ice',89.7j}
type(s)
```

Out[79]: set

In [80]:
```
# set are unordered.
s = {12,89,'snow','ice',89.7j}
print(s)
```

```
{'snow', 89, 89.7j, 12, 'ice'}
```

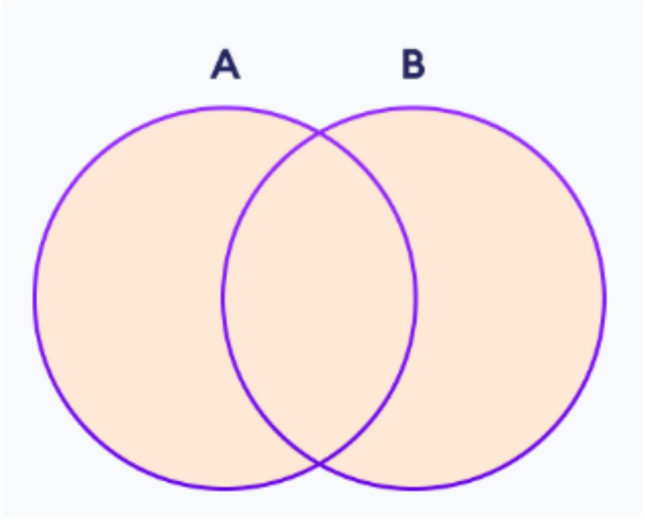
As set is an unordered sequence we cannot perform slicing and indexing in it.

In [81]:
```
# set doesnot allow duplicate values.
s = {12,89,'snow','ice',89+7j,'snow','snow'}
s
```

Out[81]: {(89+7j), 12, 89, 'ice', 'snow'}

Set Attributes :

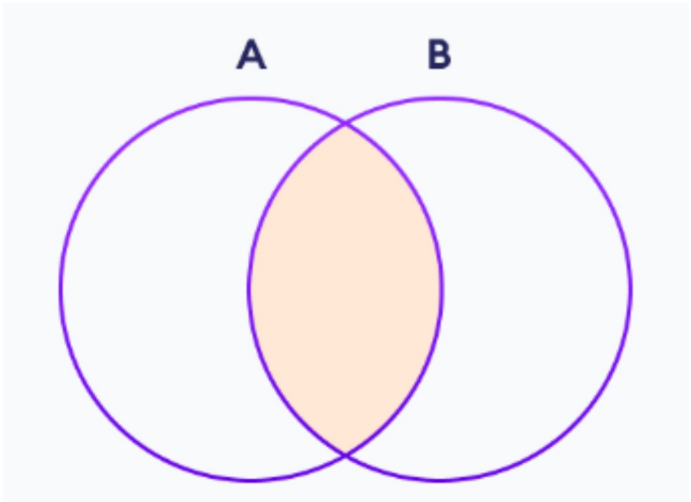
1. Union of Two Sets : The union of two sets A and B include all the elements of set A and B.



In [82]:
```
s1 = {1,2,3,4,5,6}
s2 = {'a','b','c'}
print(s1.union(s2))
```

```
{1, 2, 3, 4, 5, 6, 'c', 'b', 'a'}
```

2. Set Intersection : The intersection of two sets A and B include the common elements between set A and B.



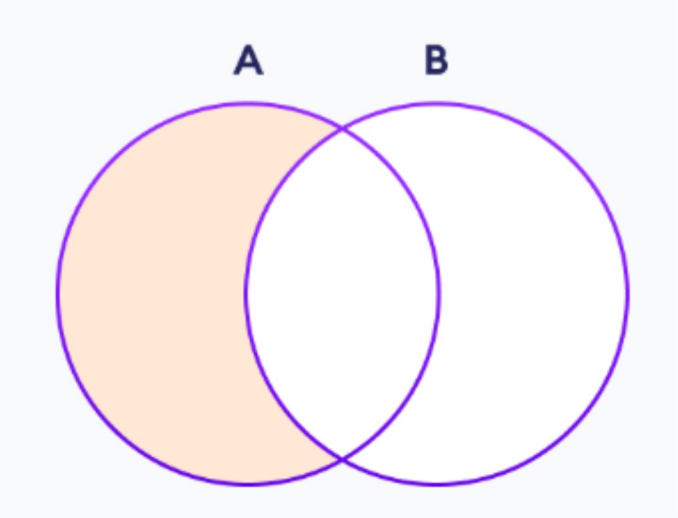
In [83]:
```
s1 = {1,2,3,4,5,6}
s2 = {'a','b','c'}
print(s1.intersection(s2))
```

```
set()
```

In [84]:
```
s1 = {1,2,3,4,5,6}
s2 = {'a','b','c',3,4}
print(s2.intersection(s1))
```

```
{3, 4}
```

3. Difference : The difference between two sets A and B include elements of set A that are not present on set B.



In [85]:
```
s1 = {1,2,3,4,5,6}
s2 = {'a','b','c',1,2,3}
print(s1.difference(s2))
```

```
{4, 5, 6}
```

In [86]:
```
s1 = {1,2,3,4,5,6}
s2 = {'a','b','c',1,2,3}
print(s2.difference(s1))
```

```
{'a', 'c', 'b'}
```

4. isdisjoint() Returns True if two sets have a null intersection
5. issubset() Returns True if another set contains this set
6. issuperset() Returns True if this set contains another set

In [87]:
```
# isdisjoint():
s1 = {1,2,3,4,5,6}
s2 = {'a','b','c',1,2,3}
s1.isdisjoint(s2)
```

Out[87]: False

In [88]:
```
s1 = {1,2,3,4,5,6}
s2 = {'a','b','c'}
s2.isdisjoint(s1)
```

Out[88]: True

In [89]:
```
# issubset():
s1 = {1,2,3,4,5,6}
s2 = {1,2,3}
```

```
s2.issubset(s1)
```

Out[89]: True

In [90]:
```
# issuperset():
s1 = {1,2,3,4,5,6}
s2 = {1,2,3}
s1.issuperset(s2)
```

Out[90]: True

In [91]:
```
# add(element):
s1 = {1,2,3,4,5,6}
s1.add('tiger')
s1
```

Out[91]: {1, 2, 3, 4, 5, 6, 'tiger'}

In [92]:
```
s1
```

Out[92]: {1, 2, 3, 4, 5, 6, 'tiger'}

In [93]:
```
# remove():
s1 = {1,2,3,4,5,6}
s1.remove(2)
s1
```

Out[93]: {1, 3, 4, 5, 6}

In [94]:
```
# pop():
s1 = {1,2,3,4,5,6}
s1.pop()
s1
```

Out[94]: {2, 3, 4, 5, 6}

In [95]:
```
# clear():
s1 = {1,2,3,4,5,6}
s1.clear()
s1
```

Out[95]: set()

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: