Strings

In [2]:
```
1 print(dir(str))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewa
rgs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__l
e__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce
__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__siz
eof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'cou
nt', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'inde
x', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'i
slower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join',
'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rind
ex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startsw
ith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

In [4]:
```
1 a = "123"
2 a[::]
```

Out[4]: '123'

In [5]:
```
1 a[::-1]
```

Out[5]: '321'

In [6]:
```
1 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_spec.
 |
 |  __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 the res
t 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 spac
e).
 |
 |  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', 'replace' an
d
 |       '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 kwar
gs.
 |      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 otherwise.
 |
 |      A string is alpha-numeric if all characters in the string are alpha-num
eric and
 |      there is at least one character in the string.
 |
```

```
 |  isalpha(self, /)
 |      Return True if the string is an alphabetic string, False otherwise.
 |
 |      A string is alphabetic if all characters in the string are alphabetic a
nd there
 |      is at least one character in the string.
 |
 |  isascii(self, /)
 |      Return True if all characters in the string are ASCII, False otherwise.
 |
 |      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 decima
l 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 digits a
nd there
 |      is at least one character in the string.
 |
 |  isidentifier(self, /)
 |      Return True if the string is a valid Python identifier, False otherwis
e.
 |
 |      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 lowerca
se 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 and the
re 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 printable
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 whitespace a
nd there
 |      is at least one character in the string.
 |
 |  istitle(self, /)
 |      Return True if the string is a title-cased string, False otherwise.
 |
 |      In a title-cased string, upper- and title-case characters may only
 |      follow uncased characters and lowercase characters only cased ones.
 |
 |  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 upperca
se 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 given str
ing.
 |      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 spac
e).
 |
 |  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 instead.
 |
 |  partition(self, sep, /)
 |      Partition the string into three parts using the given separator.
 |
 |      This will search for the separator in the string.  If the separator is
found,
 |      returns a 3-tuple containing the part before the separator, the separat
or
 |      itself, and the part after it.
 |
 |      If the separator is not found, returns a 3-tuple containing the origina
l string
 |      and two empty strings.
 |
 |  replace(self, old, new, count=-1, /)
 |      Return a copy with all occurrences of substring old replaced by new.
```

```
 |
 |        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 occurrenc
es 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 spac
e).
 |
 |  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 before th
e
 |      separator, the separator itself, and the part after it.
 |
 |      If the separator is not found, returns a 3-tuple containing two empty s
trings
 |      and the original string.
 |
 |  rsplit(self, /, sep=None, maxsplit=-1)
 |      Return a list of the words in the string, using sep as the delimiter st
ring.
 |
 |        sep
 |          The delimiter according which to split the string.
 |          None (the default value) means split according to any whitespace,
 |          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 the fr
ont.
 |
 |  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 instead.
 |
 |  split(self, /, sep=None, maxsplit=-1)
 |      Return a list of the words in the string, using sep as the delimiter st
ring.
 |
 |      sep
 |        The delimiter according which to split the string.
 |        None (the default value) means split according to any whitespace,
 |        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 boundaries.
 |
 |      Line breaks are not included in the resulting list unless keepends is g
iven and
 |      true.
 |
 |  startswith(...)
 |      S.startswith(prefix[, start[, end]]) -> bool
 |
 |      Return True if S starts with the specified prefix, False otherwise.
 |      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 remov
e.
 |
 |      If chars is given and not None, remove characters in chars instead.
 |
 |  swapcase(self, /)
 |      Convert uppercase characters to lowercase and lowercase characters to u
ppercase.
 |
 |  title(self, /)
 |      Return a version of the string where each word is titlecased.
 |
 |      More specifically, words start with uppercased characters and all remai
ning
 |      cased characters have lower case.
 |
 |  translate(self, table, /)
 |      Replace each character in the string using the given translation table.
 |
 |        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 instance
a
 |      dictionary or list.  If this operation raises LookupError, the characte
r 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 the giv
en 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 signature.
 |
 |  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 Unicode
 |      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 resul
t.
```

In [7]:
```
1 b = "apssdc"
```

In [8]:
```
1 b.capitalize()
```

Out[8]: 'Apssdc'

In [9]:
```
1 b.casefold()
```

Out[9]: 'apssdc'

In [10]:
```
1 b = "ApSSdc"
```

In [11]:
```
1 b.casefold()
```

Out[11]: 'apssdc'

In [12]:
```
1 b.swapcase()
```

Out[12]:
```
'aPssDC'
```

List:

- list() or []
- list is mutable
- list we can use different datatypes

In [13]:
```
1 l = ["apssdc",3,9.8]
```

In [14]:
```
1 l
```

Out[14]:
```
['apssdc', 3, 9.8]
```

In [15]:
```
1 type(l)
```

Out[15]:
```
list
```

In [16]:
```
1 l[0]
```

Out[16]:
```
'apssdc'
```

In [17]:
```
1 l[0][2]
```

Out[17]:
```
's'
```

In [18]:
```
1 l[1:]
```

Out[18]:
```
[3, 9.8]
```

In [19]:
```
1 l[::-1]
```

Out[19]:
```
[9.8, 3, 'apssdc']
```

In [20]:
```
1 print(dir(list))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir_
_', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem
__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass
__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setat
tr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'c
lear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'revers
e', 'sort']
```

In [21]:
```
1 help(list)
```

```
Help on class list in module builtins:

class list(object)
 |  list(iterable=(), /)
 |
 |  Built-in mutable sequence.
 |
 |  If no argument is given, the constructor creates a new empty list.
 |  The argument must be an iterable if specified.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(...)
 |      x.__getitem__(y) <==> x[y]
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __iadd__(self, value, /)
 |      Implement self+=value.
 |
 |  __imul__(self, value, /)
 |      Implement self*=value.
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
```

```
|  __lt__(self, value, /)
|      Return self<value.
|
|  __mul__(self, value, /)
|      Return self*value.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __reversed__(self, /)
|      Return a reverse iterator over the list.
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  __setitem__(self, key, value, /)
|      Set self[key] to value.
|
|  __sizeof__(self, /)
|      Return the size of the list in memory, in bytes.
|
|  append(self, object, /)
|      Append object to the end of the list.
|
|  clear(self, /)
|      Remove all items from list.
|
|  copy(self, /)
|      Return a shallow copy of the list.
|
|  count(self, value, /)
|      Return number of occurrences of value.
|
|  extend(self, iterable, /)
|      Extend list by appending elements from the iterable.
|
|  index(self, value, start=0, stop=9223372036854775807, /)
|      Return first index of value.
|
|      Raises ValueError if the value is not present.
|
|  insert(self, index, object, /)
|      Insert object before index.
|
|  pop(self, index=-1, /)
|      Remove and return item at index (default last).
|
|      Raises IndexError if list is empty or index is out of range.
|
|  remove(self, value, /)
|      Remove first occurrence of value.
|
|      Raises ValueError if the value is not present.
|
```

```
|  reverse(self, /)
|      Reverse *IN PLACE*.
|
|  sort(self, /, *, key=None, reverse=False)
|      Stable sort *IN PLACE*.
|
|  ----------------------------------------------------------------------
|  Static methods defined here:
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  ----------------------------------------------------------------------
|  Data and other attributes defined here:
|
|  __hash__ = None
```

In [22]:
```
1 l
```

Out[22]:
```
['apssdc', 3, 9.8]
```

In [23]:
```
1 l.append(9)
```

In [24]:
```
1 l
```

Out[24]:
```
['apssdc', 3, 9.8, 9]
```

In [25]:
```
1 l.append([23,45])
```

In [26]:
```
1 l
```

Out[26]:
```
['apssdc', 3, 9.8, 9, [23, 45]]
```

In [27]:
```
1 l[4]
```

Out[27]:
```
[23, 45]
```

In [28]:
```
1 l[4][1]
```

Out[28]:
```
45
```

In [29]:
```
1 print(dir(list))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir_
_', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem
__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass
__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setat
tr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'c
lear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'revers
e', 'sort']
```

In [30]:
```
1 l.clear()
```

In [31]:
```
1 l
```

Out[31]:
```
[]
```

In [32]:
```
1 l = [1,"alekhya",67,89,90]
```

In [33]:
```
1 l
```

Out[33]:
```
[1, 'alekhya', 67, 89, 90]
```

In [34]:
```
1 l.remove(67)
```

In [35]:
```
1 l
```

Out[35]:
```
[1, 'alekhya', 89, 90]
```

In [36]:
```
1 l.remove(2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-36-2110d8e9703c> in <module>
----> 1 l.remove(2)

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

In [37]:
```
1 l.remove(l[2])
```

In [38]:
```
1 l
```

Out[38]:
```
[1, 'alekhya', 90]
```

In [39]:
```
1 l.pop(1)
```

Out[39]:
```
'alekhya'
```

In [40]:
```
1 l
```

Out[40]:
```
[1, 90]
```

In [41]:
```
1 l.pop()
```

Out[41]:
```
90
```

In [42]:
```
1 l
```

Out[42]:
```
[1]
```

In [43]:
```
l = [1,1,2,34,56,78,90]
```

In [44]:
```
l
```

Out[44]:
```
[1, 1, 2, 34, 56, 78, 90]
```

In [48]:
```
l.count(1)
```

Out[48]:
```
2
```

In [46]:
```
help(list.count)
```

```
Help on method_descriptor:

count(self, value, /)
    Return number of occurrences of value.
```

Tuple

- tuple() or ()
- tuple is immutable
- tuple can use different datatypes

In [49]:
```
t  = (4,5,6,"a")
```

In [50]:
```
t
```

Out[50]:
```
(4, 5, 6, 'a')
```

In [51]:
```
type(t)
```

Out[51]:
```
tuple
```

In [52]:
```
print(dir(tuple))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewa
rgs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__l
e__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__red
uce_ex__', '__repr__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__s
ubclasshook__', 'count', 'index']
```

In [53]:
```
t[1]
```

Out[53]:
```
5
```

In [54]:
```
t
```

Out[54]:
```
(4, 5, 6, 'a')
```

In [57]:
```
t[::-1]
```

Out[57]:
```
('a', 6, 5, 4)
```

In [58]:
```
t.index(6)
```

Out[58]:
```
2
```

In [59]:
```
t.count(5)
```

Out[59]:
```
1
```

Dictionary

- { } or dict()
- Dictionary is mutable
- key:values -- pair/item
- we can access the values by using key

In [60]:
```
d = {"fruits" : ["apple","banana","orange","watermelon","mango"],
     "vegetables" : ("carrot","tomato","cabbage")}
```

In [61]:
```
d
```

Out[61]:
```
{'fruits': ['apple', 'banana', 'orange', 'watermelon', 'mango'],
 'vegetables': ('carrot', 'tomato', 'cabbage')}
```

In [62]:
```
d[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-62-123a9cc6df61> in <module>
----> 1 d[0]

KeyError: 0
```

In [63]:
```
d["fruits"]
```

Out[63]:
```
['apple', 'banana', 'orange', 'watermelon', 'mango']
```

In [64]:
```
d["vegetables"]
```

Out[64]:
```
('carrot', 'tomato', 'cabbage')
```

In [65]:
```
type(d["fruits"])
```

Out[65]:
```
list
```

In [66]:
```
d["fruits"][3]
```

Out[66]:
```
'watermelon'
```

In [67]:
```
d["vegetables"][1]
```

Out[67]:
```
'tomato'
```

In [68]:
```
print(dir(dict))
```

```
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt_
_', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len_
_', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'cle
ar', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefaul
t', 'update', 'values']
```

In [69]:
```
d.keys()
```

Out[69]:
```
dict_keys(['fruits', 'vegetables'])
```

In [72]:
```
d.items()
```

Out[72]:
```
dict_items([('fruits', ['apple', 'banana', 'orange', 'watermelon', 'mango']),
('vegetables', ('carrot', 'tomato', 'cabbage'))])
```

In [73]:
```
d.values()
```

Out[73]:
```
dict_values([['apple', 'banana', 'orange', 'watermelon', 'mango'], ('carrot',
'tomato', 'cabbage')])
```

In [75]:
```
help(dict.values)
```

```
Help on method_descriptor:

values(...)
    D.values() -> an object providing a view on D's values
```

Sets

- { } or set()
- unique
- set is immutable and duplicate elements are removed

In [76]:
```
a = {1,2,3,1,2,3}
a
```

Out[76]:
```
{1, 2, 3}
```

In [80]:
```
a
```

Out[80]:
```
{1, 2, 3}
```

In [77]:
```
print(dir(set))
```

```
['__and__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__
iand__', '__init__', '__init_subclass__', '__ior__', '__isub__', '__iter__', '_
_ixor__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__rand
__', '__reduce__', '__reduce_ex__', '__repr__', '__ror__', '__rsub__', '__rxor_
_', '__setattr__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__x
or__', 'add', 'clear', 'copy', 'difference', 'difference_update', 'discard', 'i
ntersection', 'intersection_update', 'isdisjoint', 'issubset', 'issuperset', 'p
op', 'remove', 'symmetric_difference', 'symmetric_difference_update', 'union',
'update']
```

Functions

- 2types
  - Builtin functions
  - user defined functions

In [81]:
```
import builtins
```

In [82]:
```
1 print(dir(builtins))
```

```
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'Block
ingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessErr
or', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'Co
nnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'Environment
Error', 'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'Floating
PointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'Impor
tWarning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryE
rror', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'ModuleNo
tFoundError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'Not
ImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'Pe
rmissionError', 'ProcessLookupError', 'RecursionError', 'ReferenceError', 'Reso
urceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIter
ation', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabErro
r', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeErr
or', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWar
ning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionErr
or', '__IPYTHON__', '__build_class__', '__debug__', '__doc__', '__import__', '_
_loader__', '__name__', '__package__', '__spec__', 'abs', 'all', 'any', 'asci
i', 'bin', 'bool', 'breakpoint', 'bytearray', 'bytes', 'callable', 'chr', 'clas
smethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'di
r', 'display', 'divmod', 'enumerate', 'eval', 'exec', 'filter', 'float', 'forma
t', 'frozenset', 'get_ipython', 'getattr', 'globals', 'hasattr', 'hash', 'hel
p', 'hex', 'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'li
cense', 'list', 'locals', 'map', 'max', 'memoryview', 'min', 'next', 'object',
'oct', 'open', 'ord', 'pow', 'print', 'property', 'range', 'repr', 'reversed',
'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'su
per', 'tuple', 'type', 'vars', 'zip']
```

In [83]:
```
1 help(sum)
```

```
Help on built-in function sum in module builtins:

sum(iterable, start=0, /)
    Return the sum of a 'start' value (default: 0) plus an iterable of numbers

    When the iterable is empty, return the start value.
    This function is intended specifically for use with numeric values and may
    reject non-numeric types.
```

In [84]:
```
1 a = 20
2 b = 30
3 sum((a,b))
```

Out[84]: 50

In [85]:
```
1 ord("a")
```

Out[85]: 97

In [86]:
```
chr(97)
```

Out[86]: 'a'

In [87]:
```
a = [1,2,3,4]
min(a)
```

Out[87]: 1

In [89]:
```
max(a)
```

Out[89]: 4

User defined functions

- def function_name():
  - statements
- function_name()

- types:
  - positional argumets
  - keyword arguments
  - default arguments
  - variable length arguments
  - keyword length arguments

In [90]:
```
def add():
    c = 2+9
    return c

```

In [91]:
```
add()
```

Out[91]: 11

In [92]:
```
# positional arguments --- pass in an order

def student_details(marks,name,rollnumber):
    return(marks,name,rollnumber)
```

In [95]:
```
student_details(89,"alekhya","23456789")
```

Out[95]: (89, 'alekhya', '23456789')

In [96]:
```
# keyword arguments --- key = value

def student(marks,name):
    return(marks,name)
```

In [97]:
```
student(name="alekhya",marks=90)
```

Out[97]:
```
(90, 'alekhya')
```

In [101]:
```
# default arguments

def student(marks,name,collegename ="vvit"):
    return(marks,name,collegename)
student(78,"a")
```

Out[101]:
```
(78, 'a', 'vvit')
```

In [102]:
```
student(67,"b","mvgr")
```

Out[102]:
```
(67, 'b', 'mvgr')
```

In [103]:
```
# Variable length arguments
def variable_length(*a):
    return a
variable_length(1,2,3,4,5,6,7,45,67,90,"apssdc")

```

Out[103]:
```
(1, 2, 3, 4, 5, 6, 7, 45, 67, 90, 'apssdc')
```

In [104]:
```
# keyword length arguments
def workshops(**h):
    return h
workshops(workshop1 = "DA",workshop2 = "ML",count = 90)
```

Out[104]:
```
{'workshop1': 'DA', 'workshop2': 'ML', 'count': 90}
```

Packages and Modules

- Module:
  - Module is a single python file,which contains python code
  - code represents functions,classes and variables
- Package:
  - Package is a collection of modules

In [105]:
```
import math
```

In [106]:
```
1 print(dir(math))
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acos
h', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cos
h', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floo
r', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfini
te', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'mod
f', 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan',
'tanh', 'tau', 'trunc']
```

In [108]:
```
1 math.pi # module_name.function_name
```

Out[108]: 3.141592653589793

In [109]:
```
1 math.pow(2,3)
```

Out[109]: 8.0

In [110]:
```
1 from math import *
```

In [112]:
```
1 pi # function_name
```

Out[112]: 3.141592653589793

In [113]:
```
1 import math as m
```

In [114]:
```
1 m.pi
```

Out[114]: 3.141592653589793

In [115]:
```
1 import calendar
```

In [116]:
```
1 print(dir(calendar))
```

```
['Calendar', 'EPOCH', 'FRIDAY', 'February', 'HTMLCalendar', 'IllegalMonthErro
r', 'IllegalWeekdayError', 'January', 'LocaleHTMLCalendar', 'LocaleTextCalenda
r', 'MONDAY', 'SATURDAY', 'SUNDAY', 'THURSDAY', 'TUESDAY', 'TextCalendar', 'WED
NESDAY', '_EPOCH_ORD', '__all__', '__builtins__', '__cached__', '__doc__', '__f
ile__', '__loader__', '__name__', '__package__', '__spec__', '_colwidth', '_loc
ale', '_localized_day', '_localized_month', '_spacing', 'c', 'calendar', 'datet
ime', 'day_abbr', 'day_name', 'different_locale', 'error', 'firstweekday', 'for
mat', 'formatstring', 'isleap', 'leapdays', 'main', 'mdays', 'month', 'month_ab
br', 'month_name', 'monthcalendar', 'monthlen', 'monthrange', 'nextmonth', 'prc
al', 'prevmonth', 'prmonth', 'prweek', 'repeat', 'setfirstweekday', 'sys', 'tim
egm', 'week', 'weekday', 'weekheader']
```

In [117]:
```
1 calendar.isleap(2005)
```

Out[117]: False

In [118]:
```
1 calendar.isleap(2012)
```

Out[118]:
```
True
```

In [119]:
```
1 help(calendar.month)
```

```
Help on method formatmonth in module calendar:

formatmonth(theyear, themonth, w=0, l=0) method of calendar.TextCalendar instan
ce
    Return a month's calendar string (multi-line).
```

In [121]:
```
1 print(calendar.month(2021,6))
```

```
     June 2021
Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

In [122]:
```
1 import programming
```

In [124]:
```
1 print(dir(programming))
```

```
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__package__', '__spec__', 'add', 'sub']
```

In [125]:
```
1 help(add)
```

```
Help on function add in module __main__:

add()
```

In [126]:
```
1 programming.add(5,7)
```

Out[126]:
```
12
```

In [127]:
```
1 programming.sub(90,7)
```

Out[127]:
```
83
```

In [128]:
```
1 import mypack
```

In [130]:
```
1 dir(mypack)
```

Out[130]:
```
['__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__']
```

In [132]:
```
1 from mypack import basic
```

In [133]:
```
1 dir(basic)
```

Out[133]:
```
['__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'even']
```

In [134]:
```
1 from mypack import *
```

In [135]:
```
1 from mypack import basic as b
```

In [136]:
```
1 help(b)
```

```
Help on module mypack.basic in mypack:

NAME
    mypack.basic

FUNCTIONS
    even(a)

FILE
    c:\users\alekhya\desktop\machinelearning(7-06-21)\day3\mypack\basic.py
```

In [137]:
```
1 b.even(8)
```

Out[137]:
```
True
```

In [138]:
```
1 b.even(5)
```

Out[138]:
```
False
```

File Handling

- file:
  - to store some data
- file handling
  - used to do some operations
- open
  - open(filename,mode)
  - with open(filename,mode)
    - read,write,close,append
    - close()
    - read mode - "r"
    - write mode- "w"
    - append mode - "a"

In [ ]:
```
1
```