In [1]:
```
i=0
```

In [3]:
```
while i<3:
    if i==1:
        break # exits from loop
    else:
        print(i)
        i=i+1
```

```
0
```

In [4]:
```
i=0
while i<3:
    if i==1:
        i=i+1
        continue # skips the current iteration & proceeds with next iteration
    else:
        print(i)
        i=i+1
```

```
0
2
```

In [5]:
```
i=0
while i<5:
    print("Hello...{}".format(i))
    i=i+1
else:
    print("Thankyou !!!")
```

```
Hello...0
Hello...1
Hello...2
Hello...3
Hello...4
Thankyou !!!
```

In [6]:
```
for var in "abcd":
    print("Hello",var)
else:
    print("Thankyou")
```

```
Hello a
Hello b
Hello c
Hello d
Thankyou
```

In [12]:
```
#task
s="12345678"
#how to calculate sum of digits
#display total value at end

t=0
for var in s:
    t=t+ int(var)
else:
    print("Sum of {} is {}".format(s,t))
```

```
Sum of 12345678 is 36
```

In [ ]:
```
s---> collection of chars
s= " p   y   t   h   o   n"
     |
     0   1   2   3   4   5   positive indexing

    -6   -5 -4   -3  -2 -1   negative indexing
```

In [9]:
```
#how to access a single char from a str
s="python"
print(s[0])
print(s[-6])
print(s[5])
print(s[-1])
```

```
p
p
n
n
```

In [13]:
```
s="Python programming"
#  012
print(s[1:3]) # stringname[n:m]
```

```
yt
```

In [16]:
```
print(s[2:8])
print(s[100:110])
```

```
thon p

```

In [23]:
```
print(len(s))
print(s[2:]) # reaches end of str
print(s[:5])# strts from 0th index
```

```
18
thon programming
Pytho
```

In [22]:
```
print(s[:])
print(s[-4:]) # last 4 chara
```

```
Python programming
ming
```

In [24]:
```
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 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 space).
|
|  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' 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 otherwise.
|
|      A string is alpha-numeric if all characters in the string are alpha-numeric 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 and 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 decimal 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 and there
|      is at least one character in the string.
|
|  isidentifier(self, /)
|      Return True if the string is a valid Python identifier, False otherwise.
|
|      Call keyword.iskeyword(s) to test whether string s is a reserved identifier,
|      such as "def" or "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 lowercase 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 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 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 and 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 uppercase 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 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 space).
|
|  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 separator
|      itself, and the part after it.
|
|      If the separator is not found, returns a 3-tuple containing the original 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 occurrences 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 space).
|
|  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 the
|      separator, the separator itself, and the part after it.
|
|      If the separator is not found, returns a 3-tuple containing two empty strings
|      and the original string.
|
|  rsplit(self, /, sep=None, maxsplit=-1)
|      Return a list of the words in the string, using sep as the delimiter string.
|
|        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 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 instead.
|
|  split(self, /, sep=None, maxsplit=-1)
|      Return a list of the words in the string, using sep as the delimiter string.
|
|      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 given 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 removed.
|
|      If chars is given and not None, remove characters in chars instead.
|
|  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 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 character 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 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 signature.
|
|  maketrans(...)
|      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 result.
```

In [ ]:
```
obj.methodname() # method call    "".format()

functionName()  # function call  len() print() input()
```

In [29]:
```
s="\tpython\n"
print(s.title())
print(s.upper())
```

```
print(s.lower())
```

```
        Python

        PYTHON

        python

python
```

In [32]:
```
s=" 101:x:bin:bash\n"
s.strip()# remove space,\n,\t  leadin and trailling
```

Out[32]:
```
'101:x:bin:bash'
```

In [33]:
```
s=" 101:x:bin:bash\n"
s.rstrip()
```

Out[33]:
```
' 101:x:bin:bash'
```

In [34]:
```
s=" 101:x:bin:bash\n"
s.lstrip()
```

Out[34]:
```
'101:x:bin:bash\n'
```

In [35]:
```
help([])
```

```
Help on list object:

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)
|      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.
|
|  ----------------------------------------------------------------------
|  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 [39]:
```
L=['a','b','c','d'] # List
type(L)
len(L) # no of elmts in list
```

Out[39]:
```
4
```

In [42]:
```
print(L[0])
print(L[3])
print(L[8])# index error
```

```
a
d
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-42-4fc2aeeba0e5> in <module>
      1 print(L[0])
      2 print(L[3])
----> 3 print(L[8])

IndexError: list index out of range
```

In [44]:
```
print(L[-1]) # last elemet from list
print(L[1:3])# ListName[n:m]- slicing
```

```
d
['b', 'c']
```

In [48]:
```
L=[1,2,3,4]# empty list creation
L[0]=10
L[1]=20
L[2]=30
print(L)
```

```
[10, 20, 30, 4]
```

In [52]:
```
for var in L:
    print("hello",var)
```

```
hello 10
hello 20
hello 30
hello 4
```

In [55]:
```
L=[]
L.append('x')
print(L)
L.append('y')
print(L)
L.append('z')
print(L)
```

```
['x']
['x', 'y']
['x', 'y', 'z']
```

In [56]:
```
L.insert(1,'a')
print(L)
```

```
['x', 'a', 'y', 'z']
```

In [58]:
```
L=['D1',123,1.2,True]
L[9]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-58-15083745251a> in <module>
      1 L=['D1',123,1.2,True]
----> 2 L[9]

IndexError: list index out of range
```

In [ ]:
```
Task
====

Write a python program

step 1. create a empty list
step 2. display total no.of items
step 3. using while loop (max limit of 5)
                - read a input hostname from <STDIN>
                - append the input hostname to existing list
step 4. displat the total no. of items

step 5. use for loop to iterate list of items.
```

In [60]:
```
host=[] #empty list
print("Size of host:{}".format(len(host)))
c=0
```

```
while(c<5):
    h=input("Enter the Hostname :") #read data from <STDIN>
    host.append(h) # adding new item to existing list
    c=c+1
```

```
Size of host:0
Enter the Hostname :host01
Enter the Hostname :host02
Enter the Hostname :host03
Enter the Hostname :host04
Enter the Hostname :host05
```

In [61]:
```
for var in host:
    print(var)
else:
    print("Size of host:{}".format(len(host)))
```

```
host01
host02
host03
host04
host05
Size of host:5
```

In [62]:
```
help([])
```

```
Help on list object:

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)
|      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.
|
|  ----------------------------------------------------------------------
|  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 [64]:
```
L=[1,1,2,2,3,3]
L.index(1)
```

Out[64]: 0

print(L)

In [74]:
```
L=[1,4,2,4,1,2,9]
print(L)
L.remove(1) # remove by value- removed 1 at index pos 0
print(L)
L.pop()
print(L)# remove last elemt
```

```
[1, 4, 2, 4, 1, 2, 9]
[4, 2, 4, 1, 2, 9]
[4, 2, 4, 1, 2]
```

In [75]:
```
L=[1,2,3,1,4,5,1,6,7,1,2]
L.count(1)
```

Out[75]: 4

In [76]:
```
L=["a","b","Data"]
if L.count("Data")==0:
    print("There no Data")
else:
    print("Yes Data exist at {}".format(L.index("Data")))
```

```
Yes Data exist at 2
```

In [77]:
```
# membership operator in
# a in "abcd"
s="sales"
"sales" in " 101,raj,sales,Bangalore,2121212"
```

Out[77]:
```
True
```

In [78]:
```
"sales"==" 101,raj,sales,Bangalore,2121212"
```

Out[78]:
```
False
```

In [80]:
```
"HR" not in " 101,raj,sales,Bangalore,2121212"
```

Out[80]:
```
True
```

In [81]:
```
if "sales" in " 101,raj,sales,Bangalore,2121212":
    print("Yes sales dept matched")
else:
    print("Dept not matched")
```

```
Yes sales dept matched
```

In [ ]:
```
Activity 1
==========
gn Dbs=['oracle','sql','mysql','mongo']
step 1. read input db from <stdin>
step 2. use membership operator to testdb exist or not
step 3. If exists--- display YES DB <dbname> EXIST in the
                INDEX POS <index number> use index()
step 4. If not exist, use append() to add the db to list
```

In [ ]:
```
Activity 2
==========
Given List L=['0.11','4.22','6.33','7.11','3.12','2.44']
calculate the sum of List
```

In [83]:
```
#Activity 1 Soln
Dbs=['oracle','sql','mysql','mongo']
v= input("Enter a database")
if v in Dbs:
    print("YES DB {} exists at INDEX POS {}".format(v,Dbs.index(v)))
else:
    Dbs.append(v)

for var in Dbs:
    print(var)
```

```
Enter a databaseplsql
oracle
sql
mysql
mongo
plsql
```

In [84]:
```
#Activity 2 soln
L=['0.11','4.22','6.33','7.11','3.12','2.44']
```

```
c=0
for v in L:
    c= c+ float(v)
else:
    print("Sum of List :{} ".format(c))
```

```
Sum of List :23.330000000000002
```

In [85]:
```
config_file=("/etc/passwd","/etc/pam.d","/etc/modules")
```

In [87]:
```
type(config_file)
help(())
```

```
Help on tuple object:

class tuple(object)
 |  tuple(iterable=(), /)
 |
 |  Built-in immutable sequence.
 |
 |  If no argument is given, the constructor returns an empty tuple.
 |  If iterable is specified the tuple is initialized from iterable's items.
 |
 |  If the argument is a tuple, the return value is the same object.
 |
 |  Built-in subclasses:
 |      asyncgen_hooks
 |      UnraisableHookArgs
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(self, key, /)
 |      Return self[key].
 |
 |  __getnewargs__(self, /)
 |
 |  __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.
|
|  __mul__(self, value, /)
|      Return self*value.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __rmul__(self, value, /)
|      Return value*self.
|
|  count(self, value, /)
|      Return number of occurrences of value.
|
|  index(self, value, start=0, stop=9223372036854775807, /)
|      Return first index of value.
|
|      Raises ValueError if the value is not present.
|
|  ----------------------------------------------------------------------
|  Static methods defined here:
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
```

In [91]:
```
print(config_file)
print(config_file[0]) # index based access- elmt at index pos 0
print(config_file[0:2])# slicing
for var in config_file:# membership operator
    print(var)
```

```
('/etc/passwd', '/etc/pam.d', '/etc/modules')
/etc/passwd
('/etc/passwd', '/etc/pam.d')
/etc/passwd
/etc/pam.d
/etc/modules
```

In [92]:
```
"etc/sysconfig" in config_file
```

Out[92]:
```
False
```

In [ ]:
```
import module
module.functioncall()-->(d1,d2...dn)
```

In [95]:
```
import sys
sys.path
```

Out[95]:
```
['C:\\Users\\Karthikeyan',
 'C:\\users\\karthikeyan\\Sub1',
 'D:\\Temp',
 'C:\\Users\\Karthikeyan',
 'C:\\Users\\Karthikeyan\\anaconda3\\python38.zip',
 'C:\\Users\\Karthikeyan\\anaconda3\\DLLs',
 'C:\\Users\\Karthikeyan\\anaconda3\\lib',
```

```
 'C:\\Users\\Karthikeyan\\anaconda3',
 '',
 'C:\\Users\\Karthikeyan\\anaconda3\\lib\\site-packages',
 'C:\\Users\\Karthikeyan\\anaconda3\\lib\\site-packages\\win32',
 'C:\\Users\\Karthikeyan\\anaconda3\\lib\\site-packages\\win32\\lib',
 'C:\\Users\\Karthikeyan\\anaconda3\\lib\\site-packages\\Pythonwin',
 'C:\\Users\\Karthikeyan\\anaconda3\\lib\\site-packages\\IPython\\extensions',
 'C:\\Users\\Karthikeyan\\.ipython']
```

In [98]:
```
config_file[0]="/etc/profile"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-98-3b8ac809eb5f> in <module>
----> 1 config_file[0]="/etc/profile"

TypeError: 'tuple' object does not support item assignment
```

In [100…
```
L= [[1,2,3],['a','b','c'], ['data',1,3.44]]
print(L[0][2])  #3
print(L[2][2]) # 3.44
```

```
3
3.44
```

In [101…
```
config_file=("/etc/passwd","/etc/pam.d","/etc/hosts")
#Here- add new config file
#delete a config file
#modify config file
#Typecast tuple   <---> list
```

In [106…
```
print(config_file)
tv=list(config_file)# typecast from tuple to list
tv.append("/etc/profile")
tv.append("/etc/sysconfig")
print(tv)
tv.pop(1)
t=tuple(tv) # typecast from list to tuple
print(t)
```

```
('/etc/passwd', '/etc/pam.d', '/etc/hosts')
['/etc/passwd', '/etc/pam.d', '/etc/hosts', '/etc/profile', '/etc/sysconfig']
('/etc/passwd', '/etc/hosts', '/etc/profile', '/etc/sysconfig')
```

In [108…
```
L= [ ("dx","dy"), (1,2), ("data1","data2"), (1.22,4.55) ]
L.append([10,20,30])
print(L)
L[0][1]="dz" # change "dx" to "dz" TypeError
```

```
[('dx', 'dy'), (1, 2), ('data1', 'data2'), (1.22, 4.55), [10, 20, 30]]
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-108-4807b39cadd5> in <module>
      2 L.append([10,20,30])
      3 print(L)
----> 4 L[0][1]="dz" # change "dx" to "dz" TypeError

TypeError: 'tuple' object does not support item assignment
```

In [113…
```
files=[["p1.c","p2.py","p3.sh","p4.java"], ("test.xml","text2.html")]
print(len(files))
print(type(files))
```

```
print(type(files[1]))
print(files[1][1])
print(files[-1][-1])
```

```
2
<class 'list'>
<class 'tuple'>
text2.html
text2.html
```

In [117…

```
L=list(files[1])
L.append("test3.xml")
files[1]=tuple(L)
print(files)
```

```
[['p1.c', 'p2.py', 'p3.sh', 'p4.java'], ('test.xml', 'text2.html', 'test3.xml')]
```

In [ ]:

```
TASK
=====
The user can have a max limit of 3 times to try.

step 1. read a pin from <stdin> .
step 2. Test the pin with PIN=1234.
step 3. If matched, give a success message and exit from loop
step 4. If not matched, display, "PIN NOT MATCHED" to monitor
step 5. if user missed 3 chances , display a message ,"YOUR PIN BLOCKED"
```

In [119…

```
pin=1234
c=0
while c<3:
    PIN=int(input("Enter the PIN : "))
    c=c+1
    if PIN == pin:
        print("Success !! Pin matched at {} time".format(c))
        break
    else:
        print("PIN NOT MATCHED.")

if pin!=PIN:
    print("Sorry your pin is blocked")
```

```
Enter the PIN : 123
PIN NOT MATCHED.
Enter the PIN : 1234
Success !! Pin matched at 2 time
```

In [ ]:

```
Task
====
Modify above code,
step 1. create a List
step 2. replace Success/ Failure message to monitor with List append operation
step 3. ask choice from user Wish to inspect the details, YES|yes
                    |
                display list of items.
```

In [120…

```
pin=1234
c=0
L=[]
while c<3:
    PIN=int(input("Enter the PIN : "))
    c=c+1
```

```
    if PIN == pin:
        L.append("Success !! Pin matched at {} time".format(c))
        break
    else:
        L.append("PIN NOT MATCHED.")

if pin!=PIN:
    print("Sorry your pin is blocked")

choice=input("Do You Wish to inspect the PIN details? yes | YES")
if choice == "yes" or choice =="YES":
    print("THE INPUT DETAILS:-")
    for var in L:
```

```
Enter the PIN : 123
Enter the PIN : 1234
Do You Wish to inspect the PIN details? yes | YESYES
THE INPUT DETAILS:-
PIN NOT MATCHED.
Success !! Pin matched at 2 time
```

In [121…
```
L
```

Out[121…
```
['PIN NOT MATCHED.', 'Success !! Pin matched at 2 time']
```

In [126…
```
for i in range(5):    # range(n)->starts from 0 to (n-1)
    print(i)
```

```
0
1
2
3
4
```

In [127…
```
for i in range(5,10):    # range(n,m)->starts from n to (m-1)
    print(i)
```

```
5
6
7
8
9
```

In [128…
```
for i in range(5,10,2):    # range(n,m,i)->starts from n to (m-1) with i incre
    print(i)
```

```
5
7
9
```

In [ ]:
```
import os
print("Current CPU LOAD BALANCE")

for in range(5):
    os.system("uptime;sleep 2")
```

In [129…
```
L=[16,50,300,5,40,110]

total=0

for v in L:
    total=total+v
```

```
print("Sum of numbers :{}".format(total))
```

```
Sum of numbers :521
```

In [130…

```
L=["host01","host02","host03","host04","host05"]

if "host03" in L:
    print("Host03 exists")
else:
    print("Host03 Not available")
```

```
Host03 exists
```

In [ ]:

```
files=[]
for v in range(5):
    var=input("Enter a filename")
    files.append(var)

print("Input file details ")
for v in files:
    print(v)
```

In [ ]:

```
osnames=["unix","Linux","Winx","Sunos"]

osnames[0]
osnames[1]
osnames[1]="ORACLE LINUX"
```

In [139…

```
s= "ram,sales,pune,3000"

#s.split(",") # split the str based on the delimeter----> returns List of str

# "ram,sales,pune,3000".split(",")===> L= ["ram","sales","pune","3000"]


emp=["ram,sales,pune,3000", "kumar,prod,chennai,5000", "Arjun,Hr,Hyderabad,8000"]

total=0
for v in emp:
    L=v.split(",")
    print(L)
   # cost=L[-1]
    name,dept,city,cost=L # multiple assignment
    print("Emp name is {} and his cost is {}".format(name,cost))
    total= total +int(cost)

print("*"*50)
print("\t Total cost :{}".format(total))
print("*"*50)
```

```
['ram', 'sales', 'pune', '3000']
Emp name is ram and his cost is 3000
['kumar', 'prod', 'chennai', '5000']
Emp name is kumar and his cost is 5000
['Arjun', 'Hr', 'Hyderabad', '8000']
Emp name is Arjun and his cost is 8000
**************************************************
         Total cost :16000
**************************************************
```

In [135…
```
#multiple assignment

name,id=10,20
```

In [138…
```
name,id=[10,20]
print(name)
print(id)
```

```
10
20
```

In [140…
```
help([])
```

```
Help on list object:

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)
```

```
|      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.
|
|  ----------------------------------------------------------------------
|  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 [142…

```
L=["x","z","c","s"]
L.sort()
print(L)
```

```
['c', 's', 'x', 'z']
```

In [144…

```
files=("p1.log","p2.log","p3.log","p4.log","p5.log")
c=1
for v in files:
    print("{}.{}".format(c,v))
    c=c+1
else:
    print("Total no. of files {}".format(len(files)))
```

```
1.p1.log
2.p2.log
3.p3.log
4.p4.log
5.p5.log
Total no. of files 5
```

In [145…

```
l=['a','b','c','d','e']
l[10:]
```

Out[145…

```
[]
```

In [146…

```
for var in ["mon","tues","wed","thr","fri"]:
    if var=="wed":
        continue
    else:
        print(var)
```

```
mon
tues
thr
fri
```

In [148…

```
var="root:x/bin/bash-123:text"

var.split("/")[-1]
'''
```

```
L=var.split("/")
L[-1]
'''
```

Out[148…
```
'bash-123:text'
```

In [151…
```
v1=str(120)
v2=input("Enter number")# returns str
print(v1+v2)
```

```
Enter number5
1205
```

In [156…
```
fobj=open("D:\\emp.csv","r")# open("filename","mode")
print(fobj.read())
```

```
101,arun,sales,pune,1000
234,vijay,prod,bglore,2000
143,arun,sales,mumbai,3000
145,vinay,HR,hyderabad,4000
455,theeb,sales,chennai,3455
783,leo,prod,bglore,5678
```

In [159…
```
fobj=open("D:\\emp.csv","r")
fobj.read()
```

Out[159…
```
'101,arun,sales,pune,1000\n234,vijay,prod,bglore,2000\n143,arun,sales,mumbai,3000\n145,v
inay,HR,hyderabad,4000\n455,theeb,sales,chennai,3455\n783,leo,prod,bglore,5678'
```

In [160…
```
fobj=open("D:\\emp.csv","r")
fobj.readlines()
```

Out[160…
```
['101,arun,sales,pune,1000\n',
 '234,vijay,prod,bglore,2000\n',
 '143,arun,sales,mumbai,3000\n',
 '145,vinay,HR,hyderabad,4000\n',
 '455,theeb,sales,chennai,3455\n',
 '783,leo,prod,bglore,5678']
```

In [161…
```
fobj=open("D:\\emp.csv") # by default read mode
fobj.read()
```

Out[161…
```
'101,arun,sales,pune,1000\n234,vijay,prod,bglore,2000\n143,arun,sales,mumbai,3000\n145,v
inay,HR,hyderabad,4000\n455,theeb,sales,chennai,3455\n783,leo,prod,bglore,5678'
```

In [164…
```
fobj=open("D:\\emp.csv","r")
L=fobj.readlines()
L[-3:] # extracting last 3 lines of file
L[:3] # extracting first 3 lines
```

Out[164…
```
['101,arun,sales,pune,1000\n',
 '234,vijay,prod,bglore,2000\n',
 '143,arun,sales,mumbai,3000\n']
```

In [166…
```
wobj= open("D:\\newfile.csv","w")
wobj.write("Single string")
wobj.write("Data")
wobj.close()
```

In [170…
```
fobj=open("D:\\emp.csv","r")
s=fobj.read()
```

```
fobj.close()

wobj=open("D:\\newfile.csv","w")
wobj.write(s)
wobj.close()


wobj=open("D:\\newfile.csv","a")
wobj.write("DATA1\n DATA2\n DATA3\n DATA4\n")
```

In [173…

```
# modify pin program- > write Success/Failure message to a file(append operation)
pin=1234
c=0
L=[]
wobj= open("D:\\pin.log","a")
while c<3:
    PIN=int(input("Enter the PIN : "))
    c=c+1
    if PIN == pin:
        wobj.write("Success !! Pin matched at {} time\n".format(c))
        break
    else:
        wobj.write("PIN NOT MATCHED.\n")

if pin!=PIN:
    print("Sorry your pin is blocked")

choice=input("Do You Wish to inspect the PIN details? yes | YES")
if choice == "yes" or choice =="YES":
    fobj= open("D:\\pin.log","r")
    print("THE INPUT DETAILS:-")
    for var in fobj.readlines(): # list -[line1,line2,line3]
        print(var.strip()) # remove \n char
```

```
Enter the PIN : 12
Enter the PIN : 34
Enter the PIN : 1234
Do You Wish to inspect the PIN details? yes | YESyes
THE INPUT DETAILS:-
PIN NOT MATCHED.PIN NOT MATCHED.Success !! Pin matched at 3 timePIN NOT MATCHED.PIN NOT
MATCHED.Success !! Pin matched at 3 time
```

In [175…

```
with open("D:\\newfile.csv","w") as wobj:
    wobj.write("Single string")
    wobj.write("Data")
```