-- coding: utf-8 -- *@author: GITAA*

Sequence data type

- Strings - Sequence of characters - " (or) '
- Tuples - Sequence of compound data - ()
- Lists - Sequence of multi-data type objects - []
- Arrays - Sequence of constrained list of objects (all objects of same datatype)

  - using array module from array package

- Dictionary- Sequence of key-value pairs - {}
- Sets - Sequence of unordered collection of unique data
- Range - Used for looping - using built-in range( )

- These can offer unique functionalities for the variables to contain and handle more than one data datatype at a time
- Supports operations such as indexing, slicing, concatenation, multiplication etc.,

Sequence object initialization

In [1]:

```
strSample   = 'learning'                    # string
```

In [2]:

```
pip install pandoc
```

```
Requirement already satisfied: pandoc in c:\users\kishore.s\anaconda3\lib
\site-packages (1.0.2)
Requirement already satisfied: ply in c:\users\kishore.s\anaconda3\lib\sit
e-packages (from pandoc) (3.11)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:

```
lstNumbers = [1, 2, 3, 3, 3, 4, 5]          # list with numbers
                                            # (having duplicate values)
print(lstNumbers)
```

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

In [4]:

```
lstSample  = [1,2,'a','sam',2]              # list with mixed data types
print(lstSample)                            # (having numbers and strings)
```

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

In [5]:

```
from array import *                          # importing array module

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

for x in arrSample: print(x)                 # printing values of array
```

```
1
2
3
4
```

The data types mentioned below can be used in creating an array of different data types. Code Python Type Min bytes ================================ 'b' int 1 'B' int 1 'u' Unicode 2 'h' int 2 'H' int 2 'i' int 2 'I' int 2 'l' int 4 'L' int 4 'f' float 4 'd' float 8

In [6]:

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

In [7]:

```
tupleSample = 1, 2, 'sample'                 # tuple packing
print(tupleSample)
```

```
(1, 2, 'sample')
```

In [8]:

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

In [9]:

```
# Creating dictionary using 'dict' keyword
dict_list  = dict([('first', 1), ('second', 2), ('four', 4)])
dict_list
```

Out[9]:

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

In [10]:

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

Out[10]:

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

In [11]:

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

print(rangeSample)

for x in rangeSample: print(x)            # print the values of 'rangeSample'
```

```
range(1, 12, 4)
1
5
9
```

Sequence data operations: Indexing

Indexing just means accessing elements. To access elements, the square brackets can be used. There are many methods to access elements in python.

index() method finds the first occurrence of the specified value and returns its position

Syntax: object.index(sub[, start[, end]]), object[index]

- Index of the element is used to access an element from ordered sequences
- The index starts from 0
- Negative indexing is used to access elements from the end of a list
- In negative indexing, the last element of a list has the index -1

String: Indexing

In [12]:

```
strSample = 'learning'          # string
```

In [13]:

```
strSample.index('l')            # to find the index of substring 'l' from the string 'l
earning'
```

Out[13]:

```
0
```

In [14]:

```
strSample.index('ning')         # to find the index of substring 'ning' from the string
'learning'
```

Out[14]:

```
4
```

In [15]:

```
strSample[7]                    # to find the substring corresponds to 8th position
```

Out[15]:

```
'g'
```

In [16]:

```
strSample[-2]                   # to find the substring corresponds to 2nd last positio
n
```

Out[16]:

```
'n'
```

In [17]:

```
strSample[-9]                   # IndexError: string index out of range
```

```
---------------------------------------------------------------------------
-
IndexError                                Traceback (most recent call las
t)
<ipython-input-17-dd2637980702> in <module>
----> 1 strSample[-9]                    # IndexError: string index out of
 range

IndexError: string index out of range
```

List: Indexing

Syntax: list_name.index(element, start, end)

In [18]:

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

In [19]:

```
lstSample.index('sam')          # to find the index of element 'sam'
```

Out[19]:

```
3
```

In [20]:

```
lstSample[2]                    # to find the element corresponds to 3rd position
```

Out[20]:

```
'a'
```

In [21]:

```
lstSample[-1]                    # to find the last element in the list
```

Out[21]:

```
2
```

Array: Indexing

In [22]:

```
from array import *              # importing array module
```

In [23]:

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

In [24]:

```
for x in arrSample: print(x)     # printing the values of 'arrSample'
```

```
1
2
3
4
```

In [25]:

```
arrSample[-3]                   # to find the 3rd last element from 'arrSample'
```

Out[25]:

```
2
```

Tuple: Indexing

In [26]:

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

In [27]:

```
tupSample.index('py')           # to find the position of the element 'py'
```

Out[27]:

```
5
```

In [28]:

```
tupSample[2]                    # to find the 3rd element of the 'tupSample'
```

Out[28]:

```
3
```

Set: Indexing

In [29]:

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

In [30]:

```
setSample[4]       # TypeError: 'set' object does not support indexing
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-30-b907ea72430f> in <module>
----> 1 setSample[4]       # TypeError: 'set' object does not support inde
xing

TypeError: 'set' object is not subscriptable
```

Dictionary: Indexing

- The Python Dictionary object provides a key: value indexing facility
- The values in the dictionary are indexed by keys, they are not held in any order

In [31]:

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

In [32]:

```
dictSample[2]                  # KeyError: 2 - indexing by values is not applicable in
dictionary
```

```
---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
<ipython-input-32-29139fb75065> in <module>
----> 1 dictSample[2]                  # KeyError: 2 - indexing by values
is not applicable in dictionary

KeyError: 2
```

In [33]:

```
dictSample[1]                  # to find the value corresponds to key 1
```

Out[33]:

```
'first'
```

In [34]:

```
dictSample['second']            # to find the value corresponds to key second
```

Out[34]:

```
2
```

range: Indexing

In [35]:

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

for x in rangeSample: print(x)  # print the values of 'rangeSample'
```

```
1
5
9
```

In [36]:

```
rangeSample.index(0)            # ValueError: 0 is not in range
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-36-6e72d566a242> in <module>
----> 1 rangeSample.index(0)            # ValueError: 0 is not in range

ValueError: 0 is not in range
```

In [37]:

```
rangeSample.index(9)            # to find index of element 1
```

Out[37]:

```
2
```

In [38]:

```
rangeSample[1]                  # given the index, returns the element
```

Out[38]:

```
5
```

In [39]:

```
rangeSample[9]                  # IndexError: range object index out of range
```

```
---------------------------------------------------------------------------
-
IndexError                                Traceback (most recent call las
t)
<ipython-input-39-d3bead8072c7> in <module>
----> 1 rangeSample[9]                  # IndexError: range object index o
ut of range

IndexError: range object index out of range
```

==========

Sequence data operations: Slicing

- The slice() constructor creates a slice object representing the set of indices specified by range(start, stop, step)
- Syntax: slice(stop), slice(start, stop, step)
- If a single parameter is passed, start and step are set to None

In [40]:

```
strSample[slice(4)]             # getting substring 'lear' from 'learning'
```

Out[40]:

```
'lear'
```

In [41]:

```
strSample[slice(1,4,2)]         # getting substring 'er'
```

Out[41]:

```
'er'
```

In [42]:

```
strSample[:]                    # learning
```

Out[42]:

```
'learning'
```

In [43]:

```
lstSample[-3:-1]                # ['a', 'sam']
```

Out[43]:

```
['a', 'sam']
```

In [44]:

```
dictSample[1:'second']          # TypeError: unhashable type: 'slice'
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-44-b4799cecd3b1> in <module>
----> 1 dictSample[1:'second']          # TypeError: unhashable type: 'sli
ce'

TypeError: unhashable type: 'slice'
```

In [45]:

```
setSample[1:2]                  # TypeError: 'set' object is not subscriptable
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-45-32bea28a7c03> in <module>
----> 1 setSample[1:2]                  # TypeError: 'set' object is not s
ubscriptable

TypeError: 'set' object is not subscriptable
```

In [46]:

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

Out[46]:

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

In [47]:

```
arrSample[1:-1]                 # array('i', [2, 3])
```

Out[47]:

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

In [48]:

```
rangeSample[:-1]                # range(1, 9, 4)
```

Out[48]:

```
range(1, 9, 4)
```

==========

Sequence data operations: Concatenation

Syntax: ','+','+='

In [49]:

```
lstSample+['py']                # [1, 2, 'a', 'sam', 2, 'py']
```

Out[49]:

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

In [50]:

```
print(strSample+' ','python')   # learning python
```

```
learning  python
```

In [51]:

```
arrSample+[50,60]               # TypeError: can only append array (not "list") to arra
y
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-51-e34d153b8d11> in <module>
----> 1 arrSample+[50,60]               # TypeError: can only append array
(not "list") to array

TypeError: can only append array (not "list") to array
```

In [52]:

```
arrSample+array('i',[50,60])    # array('i', [1, 2, 3, 4, 50, 60])
```

Out[52]:

```
array('i', [1, 2, 3, 4, 50, 60])
```

In [53]:

```
tupSample+=('th','on')
print(tupSample)                # (1, 2, 3, 4, 3, 'py', 'th', 'on')
```

```
(1, 2, 3, 4, 3, 'py', 'th', 'on')
```

In [54]:

```
setSample=setSample,24          # Converts to tuple with comma separated elements of se
t, dict, range
print(setSample)                # ({24, 'data', 'example', 87.5}, 24)
```

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

==========

Sequence data operations: Multiplication

Syntax: object*integer

In [55]:

```
lstSample*2                    # [1, 2, 'a', 'sam', 2, 1, 2, 'a', 'sam', 2]
```

Out[55]:

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

In [56]:

```
lstSample[1]*2                 # 4
```

Out[56]:

```
4
```

In [57]:

```
lstSample[2]*2                 # aa
```

Out[57]:

```
'aa'
```

In [58]:

```
tupSample[2:4]*2               # (3, 4, 3, 4) : Concatenate sliced tuple twice
```

Out[58]:

```
(3, 4, 3, 4)
```

In [59]:

```
tupSample[1]/4                 # 0.5
```

Out[59]:

```
0.5
```

In [60]:

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

Out[60]:

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

In [61]:

```
strSample*=3                   # concatenate thrice
```

In [62]:

```
print(strSample)               # learninglearninglearning
```

```
learninglearninglearning
```

In [63]:

```
rangeSample*2                    # TypeError: unsupported operand type(s) for *: 'range'
and 'int'
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-63-fd6649d3920a> in <module>
----> 1 rangeSample*2                    # TypeError: unsupported operand t
ype(s) for *: 'range' and 'int'

TypeError: unsupported operand type(s) for *: 'range' and 'int'
```

==========

END OF SCRIPT

In [ ]: