Agenda

Collections in Python

- Lists
- Tuples
- Strings
- Dictionary

Lists

- A data structure that stores an ordered collection of items in python is called a list ### Characteristics
- Mutable
- Linear data structure
- Mixed Type elements
- Variable length
- Zero based indexing ### Operations on list
- Replace
- insert
- sort
- delete
- append
- reverse

In [1]:

```
lst=[10,25.2,36,89,555]
lst
```

Out[1]:

```
[10, 25.2, 36, 89, 555]
```

In [3]:

```
ls1=[2,48,"python",56,1,5.4]
ls1
ls1.sort()
ls1
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-3-3cd14bfd7295> in <module>
      1 ls1=[2,48,"python",56,1,5.4]
      2 ls1
----> 3 ls1.sort()
      4 ls1

TypeError: '<' not supported between instances of 'str' and 'int'
```

In [30]:

```
animal=['cat','dog','elephant','lion']
animal.sort()
animal
```

Out[30]:

```
['cat', 'dog', 'elephant', 'lion']
```

In [17]:

```
mixed=[10,5.6,89,4,63.8,1]
mixed
mixed.sort()
mixed
```

Out[17]:

```
[1, 4, 5.6, 10, 63.8, 89]
```

In [34]:

```
animal.append(["Buffello","Horse"]) # append
```

In [35]:

```
animal
```

Out[35]:

```
['cat',
 'dog',
 'elephant',
 'lion',
 'Buffello',
 'Buffello',
 1000,
 2000,
 ['Buffello', 'Horse']]
```

In [36]:

```
animal.extend(["Buffello1","Horse1"])
animal
```

Out[36]:

```
['cat',
 'dog',
 'elephant',
 'lion',
 'Buffello',
 'Buffello',
 1000,
 2000,
 ['Buffello', 'Horse'],
 'Buffello1',
 'Horse1']
```

In [40]:

```
del animal[6]
animal
```

Out[40]:

```
['Buffello',
 'Buffello',
 'cat',
 'dog',
 'elephant',
 'lion',
 'Buffello1',
 'Horse1']
```

In [41]:

```
animal.sort() # Sort
animal
```

Out[41]:

```
['Buffello',
 'Buffello',
 'Buffello1',
 'Horse1',
 'cat',
 'dog',
 'elephant',
 'lion']
```

In [10]:

```
animal.insert(2,"PIG") # INsert
animal
```

Out[10]:

```
['Buffello', 'cat', 'PIG', 'dog', 'elephant', 'lion']
```

In [42]:

```
animal[1]="Rat"  # Replace
animal
```

Out[42]:

```
['Buffello', 'Rat', 'Buffello1', 'Horse1', 'cat', 'dog', 'elephant', 'lio
n']
```

In [43]:

```
animal.reverse() # reverse
animal
```

Out[43]:

```
['lion', 'elephant', 'dog', 'cat', 'Horse1', 'Buffello1', 'Rat', 'Buffell
o']
```

In [44]:

```
del animal[3] # particular element is deleted
```

In [45]:

```
animal
```

Out[45]:

```
['lion', 'elephant', 'dog', 'Horse1', 'Buffello1', 'Rat', 'Buffello']
```

In [46]:

```
del animal # Entire list will be deleted
```

In [47]:

```
animal
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-47-c311d926f855> in <module>
----> 1 animal

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

In [18]:

```
lst
```

Out[18]:

```
[10, 25.2, 36, 89, 555]
```

In [19]:

```
lst.pop() # Removes last element from the list
lst
```

Out[19]:

```
[10, 25.2, 36, 89]
```

In [20]:

```
lst.pop()
lst
```

Out[20]:

```
[10, 25.2, 36]
```

In [55]:

```
lst.append(55) # adding element at the end
lst
```

Out[55]:

```
[10, 20, 3.5, 1.3, 55]
```

In [56]:

```
lst.insert(2,55) # positional adding
```

In [57]:

```
lst
```

Out[57]:

```
[10, 20, 55, 3.5, 1.3, 55]
```

In [58]:

```
print(lst.count(55),lst.count(36),lst.count('A'),lst.count(3.5)) # The element repeated
n times
```

```
2 0 0 1
```

In [59]:

```
lst.index(55)
```

Out[59]:

```
2
```

In [60]:

```
lst.remove(55) # removing an element
lst
```

Out[60]:

```
[10, 20, 3.5, 1.3, 55]
```

Tuple

- A tuple is a collection of objects which ordered and immutable. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.
- Immutable

In [28]:

```
t=(1,45.6,"program",555)
t
```

Out[28]:

```
(1, 45.6, 'program', 555)
```

In [29]:

```
print(type(t))
```

```
<class 'tuple'>
```

Accessing Values in Tuples

In [2]:

```
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5, 6, 7 )
print("tup1[0]: ", tup1[0])
print("tup2[1:5]: ", tup2[1:5])
```

```
tup1[0]:  physics
tup2[1:5]:  (2, 3, 4, 5)
```

Updating Tuples

In [50]:

```
tup1 = (12, 34.56)
tup2 = ('abc', 'xyz')

# Following action is not valid for tuples
# tup1[0] = 100;
print(tup1)

# So let's create a new tuple as follows
tup3 = tup1 + tup2
print(tup3)
```

```
(12, 34.56)
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

In [5]:

```
tup = ('physics', 'chemistry', 1997, 2000)
print(tup)
del (tup)
print ("After deleting tup : ")
print(tup)
```

```
('physics', 'chemistry', 1997, 2000)
After deleting tup :

---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-5-b6baf99ac202> in <module>
      3 del (tup)
      4 print ("After deleting tup : ")
----> 5 print(tup)

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

In [ ]:

In [70]:

```
tp=(((10*2)+5))
tp
print(type(tp))
```

```
<class 'int'>
```

In [31]:

```
t1=(10,)
print(type(t1))
```

```
<class 'tuple'>
```

In [33]:

```
t
```

Out[33]:

```
(1, 45.6, 'program', 555)
```

In [73]:

```
t=('python','programming')
t
```

Out[73]:

```
('python', 'programming')
```

In [75]:

```
u=(10,20,555,t,('apple','banana','orange'))
u
```

Out[75]:

```
(10, 20, 555, ('python', 'programming'), ('apple', 'banana', 'orange'))
```

In [76]:

```
u[3]
```

Out[76]:

```
('python', 'programming')
```

In [78]:

```
u[3][0]
```

Out[78]:

```
'python'
```

In [37]:

```
num=()
num
```

Out[37]:

```
()
```

In [38]:

```
fruits=('banana','apple','orange','grapes')
fruits
```

Out[38]:

```
('banana', 'apple', 'orange', 'grapes')
```

In [39]:

```
fruits.sort()
fruits
```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
<ipython-input-39-9d7ad30ab8a5> in <module>
----> 1 fruits.sort()
      2 fruits

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

In [40]:

```
fruits.reverse()
fruits
```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
<ipython-input-40-5452523fca9f> in <module>
----> 1 fruits.reverse()
      2 fruits

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

In [41]:

```
fruits.append("Guava")
fruits
```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
<ipython-input-41-114c74144aad> in <module>
----> 1 fruits.append("Guava")
      2 fruits

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

In [42]:

```
fruits.add("Guava")
fruits
```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
<ipython-input-42-c72f517f3b9e> in <module>
----> 1 fruits.add("Guava")
      2 fruits

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

In [45]:

```
print(fruits)
len(fruits)
```

```
('banana', 'apple', 'orange', 'grapes')
```

Out[45]:

```
4
```

In [48]:

```
lst=[10,20,3.5,1.3]
tp1=(2.3,89,555,8.9)
lst
```

Out[48]:

```
[10, 20, 3.5, 1.3]
```

In [47]:

```
tp1
```

Out[47]:

```
(2.3, 89, 555, 8.9)
```

In [49]:

```
lst2=lst+tp1
lst2
```

```
--------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-49-cb4b154f8d71> in <module>
----> 1 lst2=lst+tp1
      2 lst2

TypeError: can only concatenate list (not "tuple") to list
```

In [79]:

```
t1=(10,20,30,40)
t2=(100,200,300,400)
t3=t1+t2
t3
```

Out[79]:

```
(10, 20, 30, 40, 100, 200, 300, 400)
```

In [50]:

```
name="Gitam"
'm' in name
```

Out[50]:

```
True
```

In [51]:

```
'm' not in name
```

Out[51]:

```
False
```

In [82]:

```
print(t1)
max(t1)
print(t2)
max(t2)
```

```
(10, 20, 30, 40)
(100, 200, 300, 400)
```

Out[82]:

```
400
```

In [83]:

```
min(t2)
```

Out[83]:

```
100
```

In [84]:

```
sum(t1)
```

Out[84]:

```
100
```

In [85]:

```
sum(t2)
```

Out[85]:

```
1000
```

Python - Strings

- Strings are amongst the most popular types in Python. We can create them simply by enclosing characters in quotes. Python treats single quotes the same as double quotes. Creating strings is as simple as assigning a value to a variable.

In [6]:

```
var1 = 'Hello World!'
var2 = "Python Programming"
```

Accessing Values in Strings

In [7]:

```
print("var1[0]: ", var1[0])
print("var2[1:5]: ", var2[1:5])
```

```
var1[0]:  H
var2[1:5]:  ytho
```

Updating Strings

In [8]:

```
var1 = 'Hello World!'
print("Updated String :- ", var1[:6] + 'Python')
```

```
Updated String :-  Hello Python
```

Triple Quotes

In [9]:

```
para_str = """this is a long string that is made up of
several lines and non-printable characters such as
TAB ( \t ) and they will show up that way when displayed.
NEWLINEs within the string, whether explicitly given like
this within the brackets [ \n ], or just a NEWLINE within
the variable assignment will also show up.
"""
print(para_str)
```

```
this is a long string that is made up of
several lines and non-printable characters such as
TAB (     ) and they will show up that way when displayed.
NEWLINEs within the string, whether explicitly given like
this within the brackets [
 ], or just a NEWLINE within
the variable assignment will also show up.
```

Built-in String Methods

- capitalize() Capitalizes first letter of string
- count(str, beg= 0,end=len(string)) Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given
- endswith(suffix, beg=0, end=len(string)) Determines if string or a substring of string (if starting index beg and ending index end are given) ends with suffix; returns true if so and false otherwise
- find(str, beg=0 end=len(string)) Determine if str occurs in string or in a substring of string if starting index beg and ending index end are given returns index if found and -1 otherwise.
- index(str, beg=0, end=len(string)) Same as find(), but raises an exception if str not found.
- isalpha() Returns true if string has at least 1 character and all characters are alphabetic and false otherwise.
- isdigit() Returns true if string contains only digits and false otherwise.
- islower() Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.
- isupper() Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise.
- join(seq) Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string.
- lstrip() Removes all leading whitespace in string.
- rstrip() Removes all trailing whitespace of string.
- max(str) Returns the max alphabetical character from the string str.
- min(str) Returns the min alphabetical character from the string str.
- swapcase() Inverts case for all letters in string.

In [53]:

```
s="welcome to pyhton programming"
print(s.capitalize())
```

```
Welcome to pyhton programming
```

In [54]:

```
s="welcome to pyhton programming"
print(s.swapcase())
```

```
WELCOME TO PYHTON PROGRAMMING
```

In [56]:

```
s="WELCOME TO PYHTON PROGRAMMING"
print(s.swapcase())
```

```
welcome to pyhton programming
```

In [60]:

```
s='gitam'
print(min(s))
print(max(s))
```

```
a
t
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

Python - Dictionary

- Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: {}.

Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

Accessing Values in Dictionary

In [13]:

```
dict = {'Name': 'Hari', 'Age': 17, 'Class': 'BTech'}
print("dict['Name']: ", dict['Name'])
print("dict['Age']: ", dict['Age'])
```

```
dict['Name']:  Hari
dict['Age']:  17
```

Updating Dictionary

In [61]:

```
dict = {'Name': 'Koushik', 'Age': 20, 'Class': 'BTECH'}
print("Before Updating")
print("dict['Age']: ", dict['Age'])
print("dict['Class']: ", dict['Class'])
dict['Age'] = 18; # update existing entry
dict['Class'] = "GITAM School of Technology"; # Add new entry
print("After updating")
print("dict['Age']: ", dict['Age'])
print("dict['Class']: ", dict['Class'])
```

```
Before Updating
dict['Age']:  20
dict['Class']:  BTECH
After updating
dict['Age']:  18
dict['Class']:  GITAM School of Technology
```

In [63]:

```
d1={1:'apple',2:'banana',3:'Orange'}
d2={4:'Grapes'}
d1.update(d2)
d1
```

Out[63]:

```
{1: 'apple', 2: 'banana', 3: 'Orange', 4: 'Grapes'}
```

In [64]:

```
d2
```

Out[64]:

```
{4: 'Grapes'}
```

In [70]:

```
dict1={1:'one',2:'two',3:'three'}
lst=[4,'four']
dict1.update([lst])
dict1
```

Out[70]:

```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

In [71]:

```
dict1.update(x=10,y=20,z=30)
dict1
```

Out[71]:

```
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 'x': 10, 'y': 20, 'z': 30}
```

Delete Dictionary Elements

In [73]:

```
dict = {'Name': 'Sachin', 'Age': 19, 'Class': 'BTECH'}
# del dict['Name'] # remove entry with key 'Name'
# dict.clear();     # remove all entries in dict

print("dict['Name']: ", dict['Name'])
print("dict['Age']: ", dict['Age'])
print("dict['Class']: ", dict['Class'])
del dict ;        # delete entire dictionary
dict
```

```
dict['Name']:  Sachin
dict['Age']:  19
dict['Class']:  BTECH
```

Out[73]:

```
dict
```

In [ ]:

```
# Example program on dictionary
```

In [12]:

```
a=[1,2,3,4]
b=['apple','ball','cat','dog']
my_dict={}
for i in range(len(a)):
    my_dict[a[i]]=b[i]
print(my_dict)
```

```
{1: 'apple', 2: 'ball', 3: 'cat', 4: 'dog'}
```

In [13]:

```
my_dict.values()
```

Out[13]:

```
dict_values(['apple', 'ball', 'cat', 'dog'])
```

In [14]:

```
my_dict.keys()
```

Out[14]:

```
dict_keys([1, 2, 3, 4])
```

In [15]:

```
my_dict[3]
```

Out[15]:

```
'cat'
```

In [16]:

```
my_dict['apple']
```

```
---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
<ipython-input-16-992d3f1913be> in <module>
----> 1 my_dict['apple']

KeyError: 'apple'
```

In [17]:

```
my_dict.get(1)
```

Out[17]:

```
'apple'
```

In [19]:

```
my_dict[5]='elephant'
my_dict
```

Out[19]:

```
{1: 'apple', 2: 'ball', 3: 'cat', 4: 'dog', 5: 'elephant'}
```

In [22]:

```
l=[1,2,3,4]
dict.fromkeys(l)
```

Out[22]:

```
{1: None, 2: None, 3: None, 4: None}
```

In [74]:

```
lst=[10,20,30,40]
dict.fromkeys(lst,'cse')
```

Out[74]:

```
{10: 'cse', 20: 'cse', 30: 'cse', 40: 'cse'}
```

set in python

Opearations on set

- Membership
- Add
- Remove
- Clear
- Intersection
- Difference
- Copy

In [24]:

```
set1={1,2,3,4,"Hello","python"}
"python" in set1
```

Out[24]:

```
True
```

In [25]:

```
6 in set1
```

Out[25]:

```
False
```

In [75]:

```
set1.add("programming")
```

In [79]:

```
set1.add("coding")
```

In [80]:

```
set1
```

Out[80]:

```
{1, 2, 3, 'Hello', 'coding', 'programming', 'python'}
```

In [81]:

```
set1.remove(3)
set1
```

Out[81]:

```
{1, 2, 'Hello', 'coding', 'programming', 'python'}
```

In [82]:

```
set1.clear()
set1
```

Out[82]:

```
set()
```

In [83]:

```
A={1,2,3,4}
B={5,1,6,4}
A^B
```

Out[83]:

```
{2, 3, 5, 6}
```

In [84]:

```
len(A)
```

Out[84]:

```
4
```

In [85]:

```
C=A.copy()
C
```

Out[85]:

```
{1, 2, 3, 4}
```

In [86]:

```
p={1,2,3,'apple','box'}
q={'hen',9,8,7}
p|q
```

Out[86]:

```
{1, 2, 3, 7, 8, 9, 'apple', 'box', 'hen'}
```

In [87]:

```
A={10,20,30,'hello'}
B={60,20,30,'hi'}
A-B
```

Out[87]:

```
{10, 'hello'}
```

In [88]:

```
A={10,20,30,'hello'}
B={60,20,30,'hi'}
B-A
```

Out[88]:

```
{60, 'hi'}
```

In [90]:

```
A.add(100000)
A
```

Out[90]:

```
{10, 100000, 20, 30, 'hello'}
```

In [ ]:

```

```