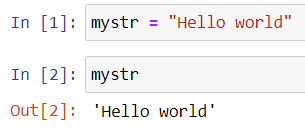
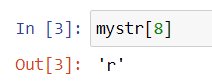
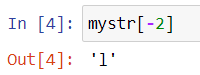
**Python Object and Data Structure Basics**

**Indexing and Slicing with Strings:**

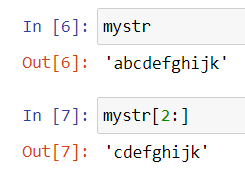
* Down we can see full string is showing.
* Now if we want a part of a string then use indexing. E.g., mystr[0] = H



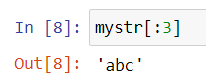
* We can also use negative indexing for this. E.g., mystr[-2] = l

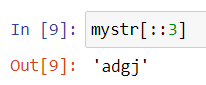


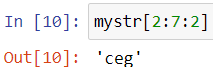
* Now we will use slicing. Format for slicing stringname[start:stop:step].
* Now we don’t mention stop then it will automatically consider it as last index.



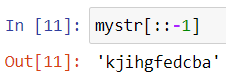
* And we don’t mention start it will consider 0th index



* Keep in mind that it will consider till the 3rd index in upper example the 3rd index in example is d which will not be printed. To print d we should use mystr[:4].
* Now we will use step also E.g., mystr[::2]. This will basically print the alternative number.
* We can use start, stop and step together.



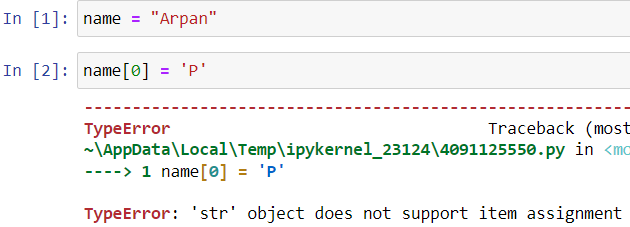
* Similarly, we can use negative in this.



* In above example if we pay attention, we can see that we have got the reversed string.

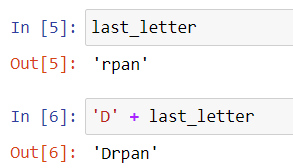
**String Properties and Methods**

* Immutable: It can’t be changed.

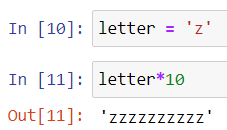


* If we want to change it the follow the process.

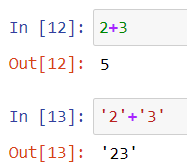




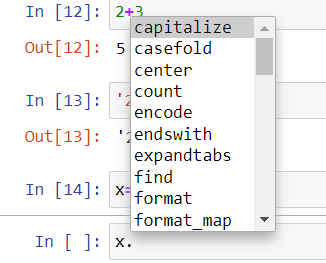
* If we use multiply sign.

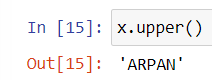


* Example downside we can see that if we add numbers it will work. But if add string it will concatenate. This is called dynamic typing so we need to be careful of datatypes.

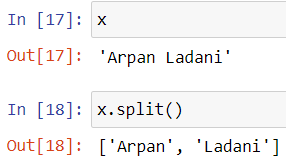


* If we x. and then press tab it will show up the all the attributes and methods.

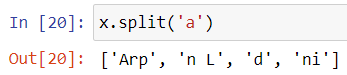




* It won’t affect the original string.
* There is a split method which will split the strings.

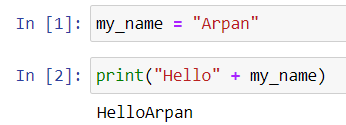


* In down example we can see that when the a appears it will split the string from there.



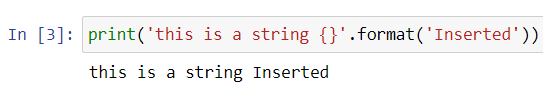
**Print Formatting with Strings**

* String interpolation is a one method to print a string with variable.

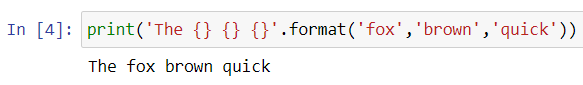


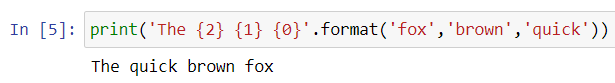
* There 2 methods of doing this .format() method and f-string (formatted string literals).
* Using .format() method syntax is

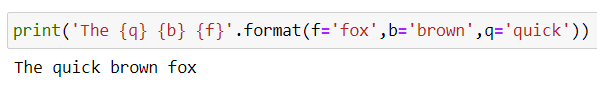
syntax: 'String here {} then also {}'.format('something1','something2')



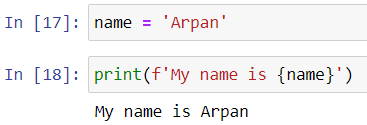
* Similarly multiple strings.



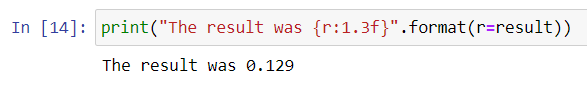
* In above example we can use indexing also.
* We can also assign a variable instead of indexing.



* Instead of writing. format() we can use f in front of string as follows.

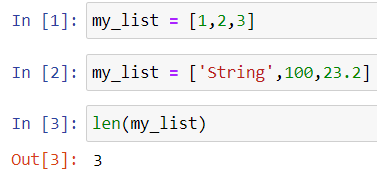


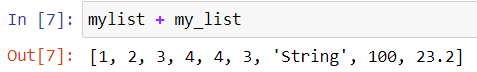
* Float formatting follows “{value:width.precision f}”
* Basically, it round of value according to precision set by us. And width is the white spaces we want before printing.



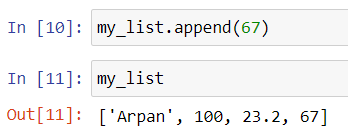
**Lists**

* Lists are ordered sequences that can hold a variety of object types.
* For notation [] are used and , to separate the objects in list.
* Lists support indexing and slicing. Lists can be nested and also have a variety of useful methods that can be called off of them.
* We can also count length of list.

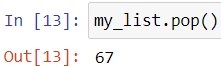




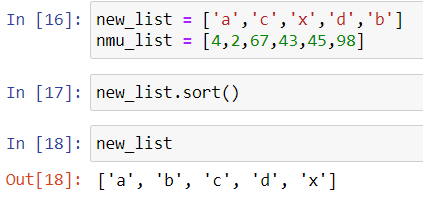
* We can change element of list by specifying index myList[0] = ‘Arpan’
* We can just add an element at end of list by using append.

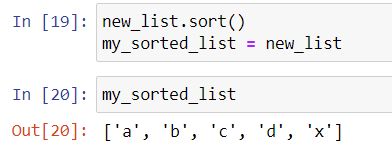


* Pop method will just pop out the last element in the list. And we can also use indexing for popping element out e.g., my\_list.pop(2)

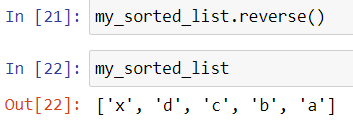


* We can use sort method for sorting the elements out in the list.



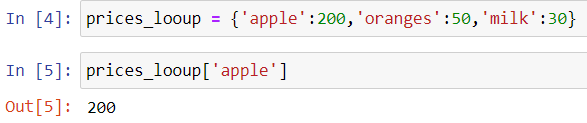


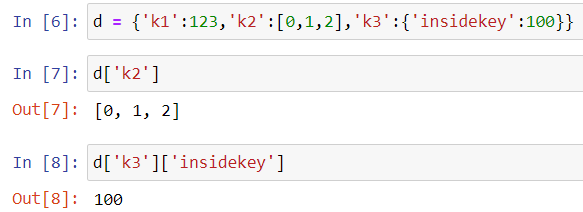
* We can reverse a list using my\_list.reverse().

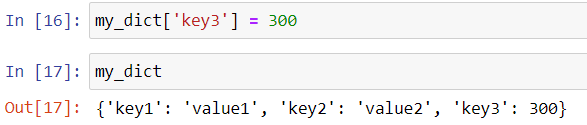


**Dictionaries**

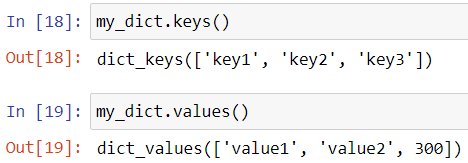
* Dictionaries are unordered mappings for storing objects. It uses a key value pairing instead.
* This key value pairing allows user to quicky grab an object that they need without the use of index.
* Syntax: {‘key1’:’value1’,’key2’:’value2’}
* Dictionaries are objects retrieved by key name and they are unordered and can’t be sorted.
* Lists are objects retrieved by location and ordered sequence can be indexed or sliced and sorted too.



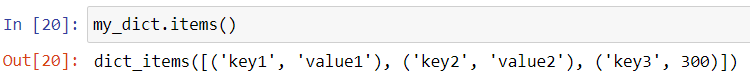
* Dictionary can also have a list in it. And we can use nested dictionary to and get objects accordingly.
* To add key value pair in dictionary. And we can change the value in particular key also.



* We can get keys and values of dictionary as follows

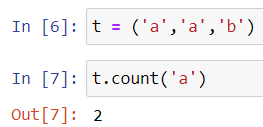


* .items() return key value pair In tuple.

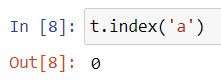


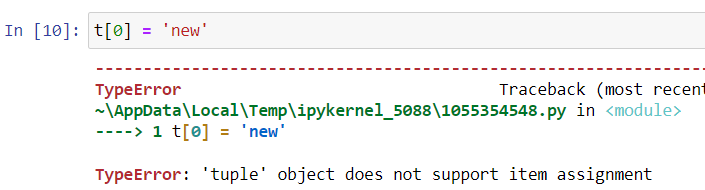
**Tuples**

* Tuples are very similar to lists. However, they have a key difference – immutability.
* Once a element is inside a tuple it can’t be changed.
* Tuples use parenthesis: (1,2,3,4)
* We can use slicing and indexing like list.
* Tuples have only 2 methods.
* One of them is count as follows



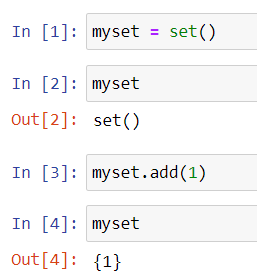
* Second is index as follows and this returns the first time when ‘a’ appears in the tuple.



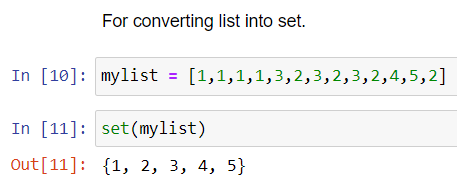
* Tuple is immutable we cannot change objects.

**Sets**

* Sets are unordered collections of unique elements.
* Meaning there can only be one representative of the same object.



* You can’t add the same element second time means all elements are different.

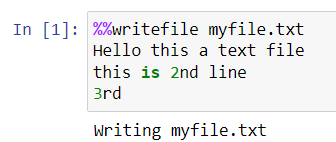


**Booleans:**

* Booleans are operators that allow you to convey True or False.
* These are imp when dealing with control flow and logic.
* And None is also part of Booleans.

**IO with Basic Files in Python:**

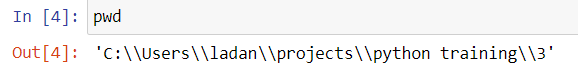
* To create a file in the same directory.

****

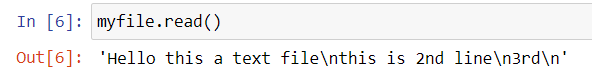
* To open a file which is there in directory.



* If not there in directory we need to give path.
* To know the working directory.



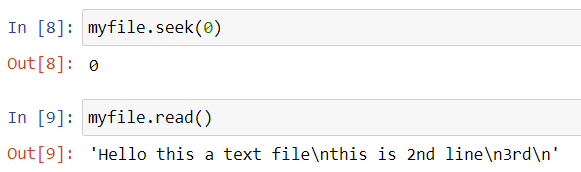
* To read the file.

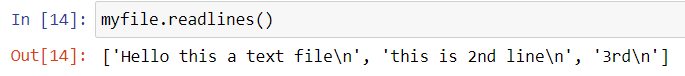
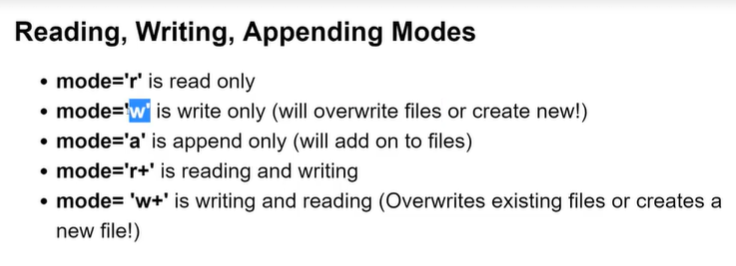


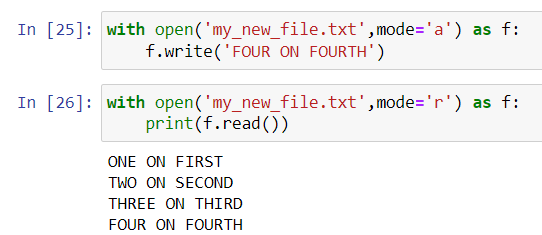
* If we run it second time this will show up



* This is because there is a cursor which reads the file. So, once we read the file the cursor is at the end of the file therefore we need to reset the cursor.



* To get the string in new line when reading.
* For reading, writing, append, etc.
* To append the text to the file.



* For write mode.

