# Python101

August 3, 2018

```
In [1]: # Simple addition
        5+98
```

```
Out[1]: 103
```

```
In [2]: #-v value addition
        -34+4
```

```
Out[2]: -30
```

```
In [3]: #Simple Substraction
        56-34
```

```
Out[3]: 22
```

```
In [4]: #Simple Division
        21/7
```

```
Out[4]: 3.0
```

```
In [5]: # Division without a floating value as answer
        21//7
```

```
Out[5]: 3
```

```
In [6]: #division with floating values
        52.777777/32.34444
```

```
Out[6]: 1.6317418697000166
```

```
In [7]: #simple multiplication
        56*4
```

```
Out[7]: 224
```

```
In [8]: #exponential function
        2**5
```

```
Out[8]: 32
```

```
In [9]: #order of operation follows PEDMAS="Parenthesis ,exponential,Division,multiplication,M
        #addition-substraction

        5**2+(2+3)*4/2

Out[9]: 35.0

In [10]: ##variables--"its a container for some values."
         my_variable=30
         my_variable2=40
         my_variable

Out[10]: 30

In [11]: my_variable2

Out[11]: 40

In [12]: #python is a case sensitive language
         My_variable


        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-12-cae254a64efc> in <module>()
          1 #python is a case sensitive language
    ----> 2 My_variable


        NameError: name 'My_variable' is not defined


In [13]: # It will throw in an error as M is in caps

In [14]: #we can perform arithmetic operations on the variables
         my_variable+my_variable2

Out[14]: 70

In [15]: # we can store this opertion in another variable
         my_addition=my_variable+my_variable2
         my_addition

Out[15]: 70

In [16]: #getting user_input
         value=input("Enter a Value:")
```

```
Enter a Value:7
```

```
In [17]: value
```

```
Out[17]: '7'
```

```
In [18]: # any value the user returns is a string so we need to change the data type to perfor
```

```
In [19]: # so what we can do is
         value=int(input("Enter a value :"))
```

```
Enter a value :7
```

```
In [20]: value+60
```

```
Out[20]: 67
```

```
In [21]: #COOL!!
```

```
In [22]: # List Tuples and Sets

         # List

         courses=["DBMS","Stats","Data Science in R","EDA"]
         print(courses)
```

```
['DBMS', 'Stats', 'Data Science in R', 'EDA']
```

```
In [23]: ## Indexing in python starts with 0

         courses[2]
```

```
Out[23]: 'Data Science in R'
```

```
In [24]: # A negetaive sign represents from the end of the list
         courses[-1]
```

```
Out[24]: 'EDA'
```

```
In [25]: courses[0:2]
```

```
Out[25]: ['DBMS', 'Stats']
```

```
In [26]: # here 0:2 means starting from 0th index  print all elements but not second index.
         # the below codes prints all the elements starting from index 1 to the end
         courses[1:]
```

```
Out[26]: ['Stats', 'Data Science in R', 'EDA']
```

In [27]:
```
# A list might contain additional list or even numeric data
my_list=[1,2,5,"foo","bar",["rock","paper","scissors"]]
my_list
```

Out[27]:
```
[1, 2, 5, 'foo', 'bar', ['rock', 'paper', 'scissors']]
```

In [28]:
```
#subetting the last element of my_list
my_list[-1]
```

Out[28]:
```
['rock', 'paper', 'scissors']
```

In [29]:
```
# it returns the last element which is a list itself
# subsetting a list within a list of my_list

my_list[-1][0:2]
```

Out[29]:
```
['rock', 'paper']
```

In [30]:
```
#adding values to list
courses.append("ML")
courses
```

Out[30]:
```
['DBMS', 'Stats', 'Data Science in R', 'EDA', 'ML']
```

In [31]:
```
# list.insert takes 2 arguments first is the index where you want to add the data , n
courses.insert(0,"Neural Networks")
courses
```

Out[31]:
```
['Neural Networks', 'DBMS', 'Stats', 'Data Science in R', 'EDA', 'ML']
```

In [32]:
```
#lets try adding a lists
courses_2=["Inferential Statistics","AI"]
courses.append(courses_2)
courses
```

Out[32]:
```
['Neural Networks',
 'DBMS',
 'Stats',
 'Data Science in R',
 'EDA',
 'ML',
 ['Inferential Statistics', 'AI']]
```

In [33]:
```
# As we can observe here list.append is just adding the whole list as a list in cours
# want to add the elements of the second list as elements we use list.extend() method
#We will use a method called list.pop() to remove the last element of the list to mak
```

In [34]:
```
courses.pop()
courses
```

```
Out[34]: ['Neural Networks', 'DBMS', 'Stats', 'Data Science in R', 'EDA', 'ML']

In [35]: courses.extend(courses_2)
         courses

Out[35]: ['Neural Networks',
          'DBMS',
          'Stats',
          'Data Science in R',
          'EDA',
          'ML',
          'Inferential Statistics',
          'AI']

In [36]: #Great
         #Now lets try removing elements from a list
         #we have already seen list.pop() in action which removes the last element of the list

         courses
         courses.remove('AI')
         courses

Out[36]: ['Neural Networks',
          'DBMS',
          'Stats',
          'Data Science in R',
          'EDA',
          'ML',
          'Inferential Statistics']

In [37]: courses.remove('Stats')

In [38]: courses

Out[38]: ['Neural Networks',
          'DBMS',
          'Data Science in R',
          'EDA',
          'ML',
          'Inferential Statistics']

In [39]: # We can see now courses dont have the elements AI and Stats

In [40]: ## Now lets see some examples of sort
         # now suppose we want to reverse the elements of the List courses
         courses.reverse()
         courses
```

```
Out[40]: ['Inferential Statistics',
          'ML',
          'EDA',
          'Data Science in R',
          'DBMS',
          'Neural Networks']

In [41]: # As we can see above the elements of courses has been reversed
         #now lets use list.sort() method
         courses.sort()
         courses

Out[41]: ['DBMS',
          'Data Science in R',
          'EDA',
          'Inferential Statistics',
          'ML',
          'Neural Networks']

In [42]: # We can see the list have been sorted alphabetically.
         # now suppose we want it (courses) to be sorted alphabetically in descending order we
         courses.sort(reverse=True)
         courses

Out[42]: ['Neural Networks',
          'ML',
          'Inferential Statistics',
          'EDA',
          'Data Science in R',
          'DBMS']

In [43]: # Cool

In [44]: # now we will use a function to short the list
         courses_sorted=sorted(courses)
         courses_sorted

Out[44]: ['DBMS',
          'Data Science in R',
          'EDA',
          'Inferential Statistics',
          'ML',
          'Neural Networks']

In [45]: # lets print out the index of elements in courses
         courses.index('EDA')

Out[45]: 3

In [46]: #lets now see if an element exists in a list (courses) or not
         'EDA' in courses
```

```
Out[46]: True

In [47]: "AI" in courses

Out[47]: False

In [48]: ### Lets see some more functions
         #lets print out the minimum value from the list
         my_list=[23,45,78,32,12,67]

         min(my_list)

Out[48]: 12

In [49]: # now  lets print maximum

         max(my_list)

Out[49]: 78

In [50]: # and now for the sum of the elements in the list

         sum(my_list)

Out[50]: 257

In [51]: # We can print the elements of a list using a for loop
         for elements in courses:
            print(elements)

Neural Networks
ML
Inferential Statistics
EDA
Data Science in R
DBMS


In [52]: # we can print the index and the elements of list using enumerate
         for index,items in enumerate(courses):
             print (index,items)

0 Neural Networks
1 ML
2 Inferential Statistics
3 EDA
4 Data Science in R
5 DBMS
```

```
In [53]: #  print from a particular index

         for index,items in enumerate(courses,start=2):
             print(index,items)
```

```
2 Neural Networks
3 ML
4 Inferential Statistics
5 EDA
6 Data Science in R
7 DBMS
```

```
In [54]: #now suppose we wana change the list into comma seperated values or seperated by any
         #we use the following code
         courses_str=', '.join(courses)
         courses_str
```

```
Out[54]: 'Neural Networks, ML, Inferential Statistics, EDA, Data Science in R, DBMS'
```

```
In [55]: courses_str='/ '.join(courses)
         courses_str
```

```
Out[55]: 'Neural Networks/ ML/ Inferential Statistics/ EDA/ Data Science in R/ DBMS'
```

```
In [56]: #convert the above string back into a list
         new_list=courses_str.split('/ ' )
         new_list
```

```
Out[56]: ['Neural Networks',
          'ML',
          'Inferential Statistics',
          'EDA',
          'Data Science in R',
          'DBMS']
```

```
In [57]: #TUPLES
         ##Tuples are very similar to lists but with one difference we cant modify tuples(Imut
         ## Lets create our first tuple

         my_tuple=('History','Geography','Math','Hindi')
         my_tuple
```

```
Out[57]: ('History', 'Geography', 'Math', 'Hindi')
```

```
In [58]: # lets change the value of the element at index 3 Hindi to sanskrit
         my_tuple[3]='Sanskrit'
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-58-a53db19d6aec> in <module>()
  1 # lets change the value of the element at index 3 Hindi to sanskrit
----> 2 my_tuple[3]='Sanskrit'


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

```
In [ ]: # what happned --- if we look closely to the error message the answer is there as earl
        #immutable so we cant change the vales or even append a value to the list, we can only
        #list
```

```
In [60]: #sets
         ##sets don't contain redundant values and reorders itself every time we print it, and
         ##so lets strt with creating a set
         my_set={'Maths','History','Geography','CompSci'}
         my_set
```

```
Out[60]: {'CompSci', 'Geography', 'History', 'Maths'}
```

```
In [62]: my_set={'Maths','History','Geography','CompSci','Maths'}
         my_set
```

```
Out[62]: {'CompSci', 'Geography', 'History', 'Maths'}
```

```
In [63]: #the last math is not printed as we already have maths element in the set
         # we can append values to a set , also subset it and use for loop to loop through it
         'Maths'in my_set
```

```
Out[63]: True
```

```
In [65]: #what we just did is checked if an element'Maths' is present in the set 'my_set' and
         #and false if not, this can be used for both list and tuple but a set is optimised fo
         #lets crete a new sets and use some opretaios on it
         my_set2={'English','History','Geography','Design'}
         my_set2
```

```
Out[65]: {'Design', 'English', 'Geography', 'History'}
```

```
In [67]: my_set.intersection(my_set2)# what it does is its find the common elements between th
```

```
Out[67]: {'Geography', 'History'}
```

```
In [68]: my_set.difference(my_set2)# here it returns the elements from the first set 'my_set'
```

```
Out[68]: {'CompSci', 'Maths'}

In [69]: my_set.union(my_set2)#here it joins both the sets keping the common values only once.

Out[69]: {'CompSci', 'Design', 'English', 'Geography', 'History', 'Maths'}

In [70]: #lets now learn to create empty list,tuple and sets
         #empty list
         empty_list=[]
         empty_list1=list()

         #empty tuple
         empty_tuple=()
         empty_tuple1=tuple()

         #set
         empty_set={}# It creates an empty dictionary not a set
         empty_set=set()

In [ ]: #Well done!! See you in the next topic till then 'May the force be with you'
```