Data strucutres / tuple, list, array, dictionary

When we have more then one variable, we can put them together. There are several ways to do it and in python there are four ways. Each data structure comes with certain capabilities. For everthing that you will do you will need a vessel to put variables in, their use in inevitable.

```
-tuples are the most basic, with least special features, they are just a buch of stuff
put together
-lists are the most general, with more features, more specific
-arrays are part of numpy - Numerial Python , and are the basic data structure type for
doing math (and more)
-dictionaries are a special type of data structures,
```

Note: Keep eye on the brackets used!

Don't forget ! use 'type()' command on a variable to check it's data strucutre type. for general check of all the defined variables type 'whos' - this will list out all the variables and their type.

```
type()
```

```
whos
```

In [14]:
```
t=(1,2,4)
```

In [15]:
```
type(t)
```

Out[15]:
```
tuple
```

In [16]:
```
l=[1,2,5]
```

In [17]:
```
type(l)
```

Out[17]:
```
list
```

In [18]:
```
whos
```

```
Variable   Type     Data/Info
-----------------------------
l          list     n=3
l2         list     n=3
t          tuple    n=3
```

```
In [19]: l2=['st','ud','ent']  # list of strings
```

```
In [20]: whos

Variable   Type     Data/Info
-----------------------------
l          list     n=3
l2         list     n=3
t          tuple    n=3
```

```
In [21]: t2=(l,l2)  # tuple of two lists
```

```
In [22]: whos

Variable   Type     Data/Info
-----------------------------
l          list     n=3
l2         list     n=3
t          tuple    n=3
t2         tuple    n=2
```

```
In [23]: l3=[t,t2]
```

```
In [24]: type(l3)

Out[24]: list
```

```
In [25]: l3 # this is how a list of tuples looks like

Out[25]: [(1, 2, 4), ([1, 2, 5], ['st', 'ud', 'ent'])]
```

```
In [26]: t2 # take a look at a tuple of lists

Out[26]: ([1, 2, 5], ['st', 'ud', 'ent'])
```

What is the difference, why do we have them different?

```
In [27]: whos

Variable   Type     Data/Info
-----------------------------
l          list     n=3
l2         list     n=3
l3         list     n=2
t          tuple    n=3
t2         tuple    n=2
```

```
In [28]: l[0]

Out[28]: 1
```

```
In [31]: t[0]

Out[31]: 1
```

Assigning elements:

```
In [32]: l[0]=5  # l is a list , we can simple change the value of a certain element
```

```
In [33]: t[0]=5   # gives error, meaning we can NOT define a new value for an existing ele

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-33-e36dcdb8325f> in <module>()
----> 1 t[0]=5

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

Notice that when calling with [] on a data strucutre, we are calling its element by its index.

Python uses numeration starting with zero. so in the list [1,2,3], the first element has index 0, the secund is with index 1, and the last element, in this list value '3' has index 2. So to say: Let's see the first element of the l list we type:

```
In [36]: l  #to display l, it's content

Out[36]: [5, 2, 5]

In [37]: l[0] # first element of l

Out[37]: 5

In [38]: l[1]

Out[38]: 2

In [40]: l[-1] # last element of l

Out[40]: 5
```

Notice that the we originaly defined l as l=[1,2,5] and that we changed the value of the first element of this LIST to '5'. l[0]=5 !

Things that you can do, capabilities of these data structure

```
Adding element (extending the list),
inserting element,
removing element,
finding element by value,
finding element by index.
retrieving index of an element with certain value.
```

```
Tuples are not extendable, no adding of elements, there is the possibility just to
combine then and to make
new tuples...
...while...
Lists are extendable, insertable, etc. - more flexible.
```

```
You can write for a list:
```

```
In [45]: l
```

```
Out[45]: [5, 2, 5]
```

```
l. <TAB>   # list name DOT and PRESS the tabulator key. Try it in the next line:
```

```
In [47]: l.
```

```
  File "<ipython-input-47-37cab99ee337>", line 1
    l.
      ^
SyntaxError: invalid syntax
```

You see append, remove, insert, etc. These are self exlpanatory...

```
In [50]: l.append(6)   #add an integer value
```

```
In [51]: l
```

```
Out[51]: [5, 2, 5, '6', 6]
```

```
In [52]: l.append('7') #add a string value
```

```
In [53]: l
```

```
Out[53]: [5, 2, 5, '6', 6, '7']
```

```
In [56]: l.insert(0,66)  # inserting integer value to the first position on the list, posi
```

```
In [57]: l
```

```
Out[57]: [66, 66, 5, 2, 5, '6', 6, '7']
```

The tuple are called immutable python objects, unchangable, while with list are flexible.

```
In [58]: #end of this notebook
In [ ]:
```