Array Concepts

In [1]:
```
#Memory of a computer stored in bits.
#unit is byte==8 bits
#Computers typically use a memory address
#Each byte is associated with a unique address. eg- Byte #1234 and Byte #2145
```

In [2]:
```
#Computer hardware is designed in such a way that any byte of the main memory can be ef
#Computer's main memory performs as RAM(Random Access Memory)
```

In [3]:
```
# as easy to retrive byte #8987409 as #568
# individual byte of memory can be stored and retrived in O(n) time
#Programming langugae keeps track of the association between an identifier and the memo
```

In [4]:
```
#A group of related variables can be stored one after another in a contiguous portion o
#This representation is an array
```

In [5]:
```
#Python internally represents each Unicode character with 16bits
#eg - 'SAMPLE' wil be like #214-215,216-217,218-219,210-211,212-213,214-215
```

In [6]:
```
#Each cell of an array uses the same number of bytes, memory addres can be calculated u
#Allows any cell to be accessed in constant time
```

In [7]:
```
#Referential Arrays-> Each element is a reference to the object which results in consta
```

In [8]:
```
#When we do list slice, we dont create new objects, we create a new reference to the ex
```

In [9]:
```
#Shallow copy for references and deep copy for creating new list from the copy module
```

Dynamic Arrays

In [10]:
```
import sys
```

In [14]:
```
n=20
data = []

for i in range (n):
    a = len(data)
    b = sys.getsizeof(data)

    print('Length: {0:3d} , Size in bytes: {1:8d}'.format(a,b))

    data.append(n)
```

```
Length:   0 , Size in bytes:       64
Length:   1 , Size in bytes:       96
Length:   2 , Size in bytes:       96
Length:   3 , Size in bytes:       96
Length:   4 , Size in bytes:       96
Length:   5 , Size in bytes:      128
Length:   6 , Size in bytes:      128
Length:   7 , Size in bytes:      128
Length:   8 , Size in bytes:      128
```

```
Length:   9 , Size in bytes:      192
Length:  10 , Size in bytes:      192
Length:  11 , Size in bytes:      192
Length:  12 , Size in bytes:      192
Length:  13 , Size in bytes:      192
Length:  14 , Size in bytes:      192
Length:  15 , Size in bytes:      192
Length:  16 , Size in bytes:      192
Length:  17 , Size in bytes:      264
Length:  18 , Size in bytes:      264
Length:  19 , Size in bytes:      264
```

In [15]:
```
#in the above, python has set no of bytes larger than needed for holding current elemen
```

Dynamic Array Implementation:

In [27]:
```
import ctypes

class DynamicArray(object):

    def __init__(self):

        self.n = 0 #Actual count of elements
        self.capacity = 1
        self.A = self.make_array(self.capacity)

    def __len__(self):

        return self.n

    def __getitem__(self,k):

        if not 0<=k<= self.n:
            return IndexError('K is out of bounds')

        return self.A[k]

    def append(self,elements):

        if self.n == self.capacity:

            self._resize(2*self.capacity) #2X if capacity is not enough

        self.A[self.n] = elements
        self.n += 1

    def _resize(self,new_cap):

        B = self.make_array(new_cap)

        for k in range (self.n):
            B[k] = self.A[k]

        self.A = B
        self.capacity = new_cap

    def make_array(self,new_cap):

        return (new_cap * ctypes.py_object)() #creating raw method
```

In [28]:
```
arr = DynamicArray()
```

In [29]:
```
arr.append(1)
```

In [30]:
```
len(arr)
```

Out[30]: 1

In [31]:
```
arr.append(2)
```

In [32]:
```
len(arr)
```

Out[32]: 2

In [33]:
```
arr[0]
```

Out[33]: 1

In [34]:
```
arr[1]
```

Out[34]: 2

In [ ]:
```

```