```
## Python Data Structure
   ###   -   LIsts
   ### -    Tuples
   ### -    Dictionaries
## Basic Problem set on Data  Structure
## Advanced Problem Set
## Packages and modules in Python###
```

In [ ]:
```
1
```

# Python Data structure

## list

In [31]:

```
li = [123,978, 654]
li # access the entir list
li[1] # access an element with index in alist

li[1:] # access all elements from second element

li = li[::-1] # access the list in reverse order

li

li = li[::-1]

li

li[::2] # accessing even index element

li[1::2]

#lists can be accessed, manipulated in two
    # direct reference - [index]
    # indirect refernce - through functions

li.append(345) # adding an element to end of the list

li

li.insert(1,234)# adding an element at a particular index

li

li.sort() # sort element in ascending order
li

li.pop()  # remove the last element in the list
li

li.pop(1) # remove the a particular element in the list
li

li2 = [234,456,789]

li.extend(li2) # merge list 2 into list 1
li

sum(li)

max(li)

len(li)

# average of list elements
sum(li)/len(li)

# average of alternate elements
sum(li[1::2])/len(li[1::2])

```

```
57
```

Out[31]: 456.0

In [43]:
```
li = [234, 1234, 6345,456,678, 912]
li
```

Out[43]: [234, 1234, 6345, 456, 678, 912]

In [46]:
```
# how to identify the second largest element in a list
    # sort the data and select the second largest number
    # sort the datain reverse order, and select
    # remove the max element and tehn get the
#li = [12, 38, 32, 78, 56]
#new_list = set(li)
#new_list.remove(max(new_list))
#print(max(new_list))

def secondlargest(li):
    li.sort()
    return li[-2]

# function that reutrns the nth largest
def genericlargest(li,n):
    li.sort()
    return li[-n]

secondlargest(li)
genericlargest(li,5)


```

Out[46]: 456

In [ ]:
```

```

In [75]:
```
# function to search for data in a list
# search for the key in the list and return the index value
def linearsearch(li, key):
    for i in range(0, len(li)):
        if li[i] == key:
            return i

linearsearch(li, 1234)

```

Out[75]: 4

In [66]:
```
def ls2(li,key):
    for element in li:
        if element == key:
            return li.index(element)
    return -1
ls2(li,912)
```

Out[66]: 3

In [67]:
```
li
```

Out[67]: [234, 456, 678, 912, 1234, 6345]

In [68]:
```
def ls3(li, key): # 3rd method of linear search
    if key in li:
        return li.index(key)
    return -1
ls3(li,678)
```

Out[68]: 2

In [71]:
```
# function to count the occurances of a character in a string
# "python programming" , m -> 2

def countchar(char, subchar):
    count = 0
    for ch in char:
        if ch == subchar:
            count += 1
    return count

#def countchar2(s,c):
 #   return s.count(c)

countchar2("python programming", "m")
```

Out[71]: 2

In [73]:
```
# function to find the number of occurance of a sub string in a string
# "abcdabcbcda", "a" -> 2
def substring(c,s):
    string.count()
 #   count = 0
  #  for ch in c:
   #     if ch == s:
    #        count += 1
    #return count


substring("abcabcdbdac", "ab")
```

Out[73]: 0

In [76]:
```
#Practice test
#1. first line contain size of the array
#second line contain arry elemnet
#       n = int(input())
s = "1 2 3 4 5 6"
li = s.split()
numberlist = []
for i in li:
    numberlist.append(int(i))
numberlist
```

Out[76]: [1, 2, 3, 4, 5, 6]

In [ ]:
```

```