**Data Structures (Veri Yapıları)**

Data (veri), işlenmemiş bilgi veya bilginin **ham** halidir. Bilgi ise, en basit anlamda verinin işlenmiş halidir. Bir programın; önemli ve biribirinde ayrılmaz bileşenleri 'veri yapıları ve algoritmadır.' Algoritma tarafından işlenen en temel elemlara (sayısal bilgiler,metinsel bilgiler, resimler, sesler,vb) veri adı verilir. Bir algoritmanın etkin, anlaşılır doğru olabilmesi için algoritmanın işleyeceği verilerin düzenlenmesi, işlenmesi gerekir. Verilerin düzenlenme biçimleride önemlidir. Çünkü yapı iyi tasarlandığında veya seçildiğinde 'etkin, anlaşılır, doğru ve hızlı çalışan ama az kaynak kullanan' programlar geliştirmek mümkün olmaktadır. Verilerin düzenlenme biçimini belirleyen yapıtaşlarına data structures denir.

Değişik alagoritmalarda verilen diziler (arrays), listler(list), demetler(tuples), kümeler(sets), sözlükler(dicts), yığınlar(stack), ve kuyruklar(queues) gibi data structures şeklinde düzenlenmesi gerekebilir. Kümerler(set), listeler(list), ve sözlükler(dict) matuble(değiştirilebir) data sturctures iken demetler(turples) ve stringler immatuble(değiştirilemez) data structures yapılıdırlar.

**dir() Fonksiyonu ile data structures'ın Metotlarını öğrenme:**

Tüm data sturucturesların kendine has metotları vardır. Hangi data structuresin hangi metotlara sahip olduğunu 'dir()' fonksiyonu ile öğrenebiliriz. örn=dir(list)

*for i in dir(list):*

*if"\_"not in i :*

*print(i,end=",")*

**List Data Structures ve Diziler:**

Python'da listeler, klasik anlamda dizilerin(arrays) yerini alan yapılardır. Fakat çok daha işelvseldir. Klasik veri yapılarından farklı olarak listlere aynı tip elemanlardan oluşmak zorunda değildir. Listeler, bir sıralı elemanlar dizisidir. [] köşli parantez arasına yazılan dizi, birer listedir ve ver itipi <list>dir.

**Listelerin Elemanlarına Erişim:**

Liste ismini takiben parantez içerisinde belirtilen numaraya indis adı verilir. Bir listenin elemanalrına aynı stringlerde olduğu gibi indis numarsı ile doğrudan ulaşılabilir ve değiştirebilirsiniz (matuble özelliğe sahiptirler). Listelerin de ilk elemanı 0 indisini alır, son indisi de -1 indisini alır.

**Listlerde Stack (Yığın) Data Structures'i:**

Stack(yığın), eleman ekleme ve çıkarma işlemlerinin dizinin listenin en son konumuna(top) göre yapıldığı özel bir yapıdır. Stack, son giren ilk çıkar **(Last In-First Out)(LIFO)** mantığı ile çalışır ve ara elemalara erişim doğrudan yapılmaz.

Stack yapısında, sadece en son eklenen (en üstteki) elemana erişilebilir. Bu özelliği itabari ile stackler, üst tarafı açık alt tarafı kapalı olan aynı zamanda (sürekli olarak büyüyen ya da küçülen) sıralı dinamik listelere benzemektedir.

Stack yapılarında iki temel işlem söz konusudur. Birinci yığına eleman ekleme (append) diğeri isie yığına en son eklenen elemanın çıkarılması (pop) işlemidir. Stack yapısına yeni bir eleman eklemek veya çıkarmak istendiğinde, bu elemanı listenin en sonuna eklemek için append(), çıkarmak için pop() metotları/fonksiyonları kullanılır.

**Listelerde (Queue) Data Structures Oluşturma:**

Queue(kuyruk), eleman ekleme işlemlerinin listenin sounundan, çıkarma işleminin listenin başından yapıldığı özel bir veri yapısıdır. Queue, ilk firen ilk çıkar **(In First-Out First)(FIFO)** mantığı ile çalışır ve ara elemalra erişim doğrudan yapılmaz.

Queue yapısında; yeni bir eleman ekleneceği zaman kuyruğun sonuna eklenir, bir elemana çıkarılacağı zaman ilk eleman çıkarılır. Queue yapsında da stack yapısında olduğu gibi iki işlem söz konusudur. Birincisi Queue yapısına eleman ekleme(append) diğeri ise queue yapısında eleman çıkarma(popleft) işlemidir. Python queue data structures'ında bir elemanı kuyruğun en sonuna eklemek için append(), kuyruğun başından çıkarmak için ise pop(0) veya popleft() metotları kullanılır. Ayrıca bu işlemler için "deque" sınıfına iat metotlar da (popleft()gibi) kullanılabilir.Bunun için 'from collections import deque' komutu satırnı programın başına eklemek gerekir. deque(double ended queue) sınıfı; oluşturduğu listenin hem başına hem de sonuna erişen metotlara sahiptir. Örneğin listenin sonuna append() ile eleman eklerken, pop() ile siler, listenin başina ise appendleft() ile eleman eklerken popleft() ile siler.

**Çok Buyutlu Liste/Dizi (Matris) Tanımlaması:**

Liste veri yapısını kullanarak tek boyutlu dizilerin (vektörlerin) tanımlanmasında tek köşeli parantez [] kullanılırken iki boyutlu dizilerin(matrislerin) tanımlanmasında iki köşeli parantez[][], üç boyutlu dizilerin tanımlanmasında üç köşeli parantez [][][] kullanılır.

**Tuple (Demet) Data Structures:**

Tuple(demetler); sıralı, değişmez, nesneler dizisidir. Genel olarak (zorunlu olmasada) parantezler arasına yazılan birbirinden virgülle ayrılmış sıralı elemanlar demetidir. Özellikleri;

* Demet elemanları doğrudan değiştirilmez (immatuble özelliği)
* Demet veri tipi; <tuple>'dır.
* Demet veri yapısındaki elemanlara indis numarsı ile erişebilir.
* Değişmez yapısı nedeni ile tuple elemanları dict (sözlük) veri yapısında anahttar (key) değer olarak kullanılabilirken listeler kullanılmaz.
* Tuple veri yapısı ile birlikte count() ve index() metotları haricinde yerleşik (builtlerin) Python fonksiyonlarını da (enumerate(), len(), max(), min(), sum(), zip(), gibi) kullanılabilir.
* Listelere göre daha hızlı çalışırlar.
* D=(1,2,3,4,5) ; T=list(D) #Tuple'dan liste'ye dönüşüm.
* L=[1,2,3,4,5] ; T=tuple(L) #Liste'den tuple'a dönüşüm.
* S="furkan" ; T=tuple(S)
* t=12,54,"kzly"
* U=t,1,2,3,4 #iç içe tuple

**Tuple elemanalrına erişim:**

Bir tuple elemanlarına aynı string ve listelerde olduğu gibi indis numaralrı ile doğrudan erişilebilir ancak değeri değiştirilmez.(immatuble özelliği) Bir tuple elemanları değiştirilmez ama tuple list veya stringe dönüştürülerek elemanlarını değiştirebiliriz.

**Liste veya Tuple Yapılarında Arama:**

Sıralı ya da sırasız bir dizideki/sıradaki herhangi bir elemanın yerinin bulunması işlemine arama denir. Bir liste ya da tuple içerisindeki veriler (sayısal ya da sözel stringler olabilir) belirli bir anahtar (key) ifadeye göre aranır. Bir A listesinde, x elemanının listede olup olmadığı veya x elemanın yerinin öğrenilmesi işlemi arama işlemi olarak tanımlanır. Python'da arama işlemin en basit yolu; 'x in A' şeklinde x elemanının A listesinde veya tuple'ında aramaktadır. For döngülerinde olduğu gibi ; in operatörü ' eğer x, A veri yapısı içinde varsa True, yoksa False' sonucunu geri döndürür.

**Set (Küme) Data Structures :**

Küme (set); sırasız, tekil elemanlar topluluğudur. Küme elemanlarının süslü {} parantezler arasına yazılır. Matematiksel küme işlemleri yanında basit ekleme ve çıkarma işlemleri vardır. Özellikleri;

* Matematiksel küme işlemlerini (birleşim,fark,kesişim) destekler.
* Bir küme içerisinde tekrarlı (aynı) elemanlar olmaz.
* Bir küme elemanları sıralı olmak zorunda değildir.
* Küme elemanlarına erişim için indis kullanılmaz.
* List, tuple data structures'larında olduğu gibi set(küme) veri yapılarına da yeni elemanlar eklenebilir. Fakat sadece elemları doğrudan değiştirilmeyen veri tipleri (tuple,str,int,float) kümeye eklenebilir.
* k={"elma","armut","mandalina"}
* k={1,2,3,4,5}
* a=set('abracadabra')
* b=set('alacazam')
* k=set()
* boş bir kümeyi tanımlarken {} yapamayız yaparsak type'ı dict olur.

**Elemanları Değiştirilmeyen Küme (frozenset):**

Bir kümenin elemlarının değiştilmesi istenmiyorsa set( ) yerine frozenset() fonksiyonu kullanılır. Böylece normal küme değiştirilmez (immatuble) özelliği kazanır. Eğer bir kümenin elemanlarının değiştirlmesi (yani küme üzerinde add(), remove(), update() gibi işlemler gerçekleştirilsin) istenmiyorsa normal küme, dondurulmuş kümeye (frozenset) dönüştürülür. frozenset() küme elemanlarını değiştirilmez (immatuble) yapar. Normalde küme elemanları değiştirilebilir (matuble) elemanlardan oluşur. Fakat bir kümeye bir eleman eklenmek istendiğinde o elemanın değiştirilmeyen (immatuble) bir veri tipinde (demet, sayı, string) olması gerekir. Dolayısyla değiştirilebilen (matuble) veri tipindeki (liste , sözlük veya küme) bir eleman, mevcut kümeye eklenmez.

**Dict (Sözlük) Data Structures :**

Sözlükler; biribirinden virgülle ayrılmış 'anahtar: değeri' / 'key:value' şeklinde eşlenen veri yapılarıdır. Buna göre; ['bir':1, 'iki':2] veya [1:'bir', 2:'iki'] veya dict sarmalına alınan her veri yapısı birer sözlüktür.

* a=dict(bir=1, iki=2, uc=3)
* b={'bir':1, 'iki':2, 'uc':3}
* c=dict(zip(['bir', 'iki', 'uc'], [1,2,3]))
* d=dict([('iki',2),('bir',1),('uc',3)])
* e=dict({'uc':3, 'bir':1, 'iki':2})
* a==b==c==d==e
* s={} # boş kümedir.

**Özellikleri;**

* Sözlük elemanları değiştirilebilir (matuble) özelliktedir.
* Birbirinden virgüllerle ayrılmış 'anahtar:değeri/'key:value' eşlem (map) yapılıdır.
* Değiştirilebilir (matuble) veri tipindeki (liste,sözlük veya küme) bir eleman, sözlük anahtar değeri olarak kullanılmaz.
* Stringler de "...", listlerde [...], tuplelarda (...), sözlüklerde ise {...} süslü parantezler kullanılır.
* String, liste ve tuplelarda elemanlara erişim için tam sayı indisler kullanılırken, dict'lerde anahtar (key) kullanılır ve her bir anahtar değeri tekil (unique) olmalıdır.