**الگوریتم چیست؟**

**الگوریتم** مجموعه‌ای متناهی از دستورالعمل‌ها است، که به ترتیب خاصی اجرا می‌شوند و مسئله‌ای را حل می‌کنند. به عبارت دیگر یک الگوریتم، روشی گام به گام برای حل مسئله است. دستورهای الگوریتم باید با زبانی دقیق، و بی‌ابهام بیان شوند. الگوریتم باید دارای شروع و پایان مشخصی باشد، به نحوی که اگر دستورهای آن را دنبال کنیم، برای تمامی حالت ها، الگوریتم پس از طی مراحل، خاتمه یابد. به علاوه، زمان لازم برای خاتمه الگوریتم هم باید به گونه‌ای معقول و کوتاه باشد. واژه الگوریتم از نام ریاضیدان و ستاره‌شناس و جغرافی‌دان نامی ایرانی، ابوجعفر محمد بن موسی خوارزمی گرفته شده‌است.

=======================================================================

# ساختارهای داده راهی برای سازماندهی و ذخیره سازی داده ها هستند. آنها رابطه بین داده ها و عملیات منطقی مختلفی را که می توان روی داده ها انجام داد توضیح می دهند.

# **ساختار داده‌های اصلی :**

اینها ساختارهای داده پایه در پایتون هستند که حاوی مقادیر داده خالص و ساده هستند و به عنوان بلوک های ساختمانی برای دستکاری داده ها عمل می‌کنند :

- **اعداد صحیح(integer):** این نوع داده برای نمایش داده های عددی، یعنی اعداد صحیح یا منفی بدون نقطه اعشار استفاده می شود. مثلا، -1، 3، یا 6.

- **اعداد اعشاری(float):** Float به معنای «عدد اعشاری» است. این عدد برای نمایش اعداد گویا استفاده می‌شود که معمولاً حاوی یک نقطه اعشار مانند 2.0 یا 5.77 هستند.

- [رشته(string)](https://www.mongard.ir/courses/python-beginner-course/episode/4/python-string-first/)**:** این نوع داده مجموعه ای از حروف الفبا، کلمات یا عدد را نشان می دهد. با گنجاندن یک سری کاراکتر در یک جفت نقل قول دوتایی یا تکی ایجاد می شود.

- [درست یا غلط(boolean)](https://www.mongard.ir/articles/31/understanding-boolean-logic-in-python/)**:** این نوع داده در مقایسه و عبارات شرطی مفید است و می تواند مقادیر TRUE یا FALSE را بگیرد.

**# **ساختارهای داده فرعی داخلی پایتون****

برخلاف ساختارهای داده اصلی، انواع داده های فرعی نه تنها مقادیر را ذخیره می کنند، بلکه مجموعه ای از مقادیر را در قالب های مختلف ذخیره می‌کنند :

- [لیست(list)](https://www.mongard.ir/courses/python-beginner-course/episode/6/python-lists/)**:** این چندمنظوره‌ترین ساختار داده در پایتون است و به‌صورت فهرستی از عناصر جدا شده با کاما در داخل براکت نوشته می‌شود. یک لیست می تواند از عناصر ناهمگن و همگن تشکیل شده باشد. [ ]

- [تاپل(tuple)](https://www.mongard.ir/courses/python-beginner-course/episode/17/python-tuple/)**:** تاپل ها شبیه لیست ها هستند اما تغییر ناپذیر هستند. همچنین برخلاف لیست ها، تاپل ها به جای براکت در داخل پرانتز مشخص می شوند. ویژگی تغییرناپذیری نشان می‌دهد که وقتی یک عنصر در یک Tuple تعریف شد، نمی‌توان آن را حذف، تخصیص مجدد یا ویرایش کرد. این تضمین میکند که مقادیر اعلام شده ساختار داده دستکاری یا نادیده گرفته نمی شوند. ( )

- [دیکشنری(dictionary)](https://www.mongard.ir/courses/python-beginner-course/episode/19/python-dictionary/)**:** دیکشنری ها از جفت های key-value تشکیل شده اند. "key" یک آیتم را مشخص می کند و "value" ارزش آن مورد را ذخیره می کند. دو نقطه key را از value آن جدا می کند. آیتم ها با کاما از هم جدا می شوند و کل آیتم ها در داخل آکولاد قرار می گیرند. { }

- [مجموعه(set)](https://www.mongard.ir/courses/python-beginner-course/episode/18/python-set/)**:** set دنباله ای نامرتب از عناصر منحصر به فرد است. مانند لیست ها، مجموعه ها قابل تغییر هستند و در داخل آکولاد نوشته می شوند، اما هیچ دو مقدار نمی توانند یکسان باشند. { }

# ساختارهای داده فرعی خاجی پایتون :

دسته دیگری از ساختارهای داده در پایتون وجود دارد که توسط کاربر تعریف شده است. یعنی کاربران آنها را تعریف می کنند:

- **پشته(stack):** پشته ها ساختارهای داده خطی در پایتون هستند. ذخیره آیتم ها در Stacks بر اساس اصول First-In/Last-Out (FILO) یا Last-In/First-Out (LIFO) است. در Stacks، افزودن یک عنصر جدید در یک انتها با حذف یک عنصر از همان انتها همراه است.

- [صف(queue)](https://www.mongard.ir/one_part/140/python-queue-module/)**:** مانند Stacks، صف ها ساختارهای داده خطی هستند. با این حال، آیتم ها بر اساس اصل First-In/ First-Out (FIFO) ذخیره می شوند. در یک صف، موردی که اخیراً حداقل اضافه شده است ابتدا حذف می شود.

- **درخت(tree):** درختان ساختارهای داده غیرخطی در پایتون هستند و از گره هایی تشکیل شده اند که توسط edge ها به هم متصل شده اند. ویژگی های یک درخت این است که یک گره به عنوان گره ریشه تعیین می شود، به غیر از ریشه، هر گره دیگری یک گره والد مرتبط دارد، و هر گره می تواند تعداد دلخواه گره فرزند داشته باشد.

- [لیست پیوندی(linked list)](https://www.mongard.ir/articles/119/linked-lists-python/)**:** مجموعه‌ای از عناصر داده‌ای که از طریق پیوندها(link) به یکدیگر متصل شده‌اند، در پایتون لیست پیوندی نامیده می‌شوند. همچنین یک ساختار داده خطی است. هر عنصر داده در یک لیست پیوندی با استفاده از اشاره گر به دیگری متصل می شود.

**# الگوریتم های پیمایش درختی :**

پیمایش(traverse) فرآیند بازدید از تمام گره های یک درخت است که از گره ریشه شروع می شود. یک درخت را می توان به سه روش مختلف طی کرد:

**- پیمایش ترتیبی(In-order)** شامل بازدید از زیر درخت در سمت چپ، و سپس زیردرخت سمتراست است.

**- در پیمایش پیش‌ سفارشی(pre-order)**، اولین موردی که بازدید می‌شود، گره ریشه و به دنبال آن زیردرخت سمت چپ و در نهایت، زیردرخت سمت راست است.

**- در الگوریتم پیمایش پس سفارشی(post-order)**، ابتدا از زیر درخت سمت چپ بازدید می شود، سپس از زیر درخت سمت راست بازدید می شود و گره ریشه آخرین بازدید می شود.

**# الگوریتم های مرتب سازی :**

الگوریتم‌های مرتب‌سازی روش‌هایی را برای چیدمان داده‌ها در یک قالب خاص نشان می‌دهند. مرتب سازی تضمین می کند که جستجوی داده ها در سطح بالایی بهینه شده و داده ها در قالب قابل خواندن ارائه می شوند. بیایید به سه نوع مختلف الگوریتم مرتب سازی در پایتون نگاه کنیم:

**- مرتب سازی حباب(bubble sort):** این الگوریتم مبتنی بر مقایسه است که در آن در صورت نادرست بودن ترتیب عناصر مجاور، به طور مکرر تعویض می شود.

**- مرتب سازی ترکیب(merge sort):** مرتب سازی Merge آرایه را به دو نیمه تقسیم می کند، آنها را مرتب می کند و سپس آنها را با هم ترکیب می کند.

**-مرتب سازی درج(insertion sort):** این الگوریتم با مقایسه و مرتب سازی دو عنصر اول شروع می شود. سپس، عنصر سوم با دو عنصر طبقه بندی شده قبلی و غیره مقایسه می شود.

**# الگوریتم های جستجو :**

الگوریتم های جستجو به بررسی و بازیابی یک عنصر از ساختارهای مختلف داده کمک می کنند. یکی از انواع الگوریتم جستجو از روش جستجوی متوالی استفاده می کند که در آن لیست به صورت متوالی پیمایش می شود و هر عنصر بررسی می شود (جستجوی خطی). در نوع دیگر، جستجوی بازه ای، عناصر در ساختارهای داده مرتب شده (جستجوی باینری) جستجو می شوند. اجازه دهید به چند نمونه نگاه کنیم:

**-** [جستجوی خطی(linear search)](https://www.mongard.ir/courses/algorithms/episode/643/algo-linear-search/)**:**در این الگوریتم هر آیتم به صورت متوالی یکی یکی جستجو می شود.

**-** [جستجوی دودویی(binary search)](https://www.mongard.ir/courses/algorithms/episode/644/algo-binary-search/)**:** در این الگوریتم فاصله جستجو بارها و بارها به نصف تقسیم می شود. اگر عنصری که باید جستجو شود کمتر از جزء مرکزی بازه باشد، فاصله به نیمه پایین‌تر کاهش می‌یابد. در غیر این صورت به نیمه بالایی باریک می شود. این فرآیند تا زمانی که مقدار مورد نظر پیدا شود تکرار می شود

=======================================================================

https://github.com/TheAlgorithms/Python

https://github.com/keon/algorithms/tree/master/algorithms

=======================================================================

**complexity :**

در علم کامپیوتر، **پیچیدگی زمانی**(**time complexity**)، پیچیدگی محاسباتی است که میزان زمان لازم برای اجرای یک الگوریتم را توصیف می کند.

نماد big O روشی برای تعیین سرعت یک الگوریتم است. با استفاده از نماد Big O، می توانیم بفهمیم که الگوریتم ما سریع است یا کند. این دانش به ما امکان می دهد الگوریتم های بهتری طراحی کنیم. همینطور باید به space complexity برای نوشتن الگوریتم ها دقت کرد

O به معنی operation هایی است که الگوریتم شما انجام میدهد و مقداری که داخل پرانتز قرار میگیرد به معنی ورودی الگوریتم شما میباشد

**# constant complexity :**پیچیدگی زمانی ثابت که با نماد O(1) نمایش داده می‌شود به این صورت که تعداد ورودی ها ثابت میباشد به همین دلیل پیچیدگی زمانی الگوریتم ثابت میباشد

**# log complexity :** که با نماد O(log n) نمایش داده می‌شود برای مواقعی است که با هربار اجرا کردن الگوریتم بازه زمانی الگوریتم نصف می‌شود ( مثلاً جست‌و‌جو در لیست مرتب شده به صورت binary search که با هربار اجرا لیست نصف می‌شود )

**# linear complexity :** که با نماد O(n) نمایش داده می‌شود به این صورت که میزان ورودی الگوریتم شما با میزان operation یا عملیات شما ارتباط مستقیم دارد ( مثلاً الگوریتم جست‌و‌جو در یک لیست نامرتب یا پیدا کردن بزرگترین عضو در یک لیست نامرتب)

**# polynomial complexity :** که با نماد O(n^n) نمایش داده می‌شود که n توان به صورت عدد میباشد ( O(n^2) ) که به تعداد حلقه های تودرتو این عدد بیشتر می‌شود . اگر ۲ حلقه به صورت جدا باشد مرتبه به صورت O(n) میباشد اما اگر یک حلقه تودرتو با ۲ حلقه باشد برابر با O(n^2) میباشد . مثلاً اگر حلقه تودرتو ما دارای ۳ حلقه باشد برابر با O(n^3) میباشد

**# exponential complexity :** که با نماد O(2^n) نمایش داده می‌شود. برای الگوریتم هایی است که قراره تمام اتفاقات ممکن را محاسبه کند . مثلاً تمام ترکیبات ممکن با حروف الفبا ( با هر بار اجرای الگوریتم بازه زمانی ۲برابر می‌شود )

=======================================================================

**limit**هدف این الگوریتم اینست که یک آرایه را محدود کند. به این شکل که شما میتوانید مقدار بیشترین و کمترین را تعیین کنید تا نتیجه براساس محدودیت‌های شما نمایش داده شود.

**Top\_one**|  
الگوریتم top one مقادیری که در یک آرایه بیشترین تکرار را داشته‌اند را برمیگرداند

**caesar cipher**این نوع رمزنگاری توسط ژولیوس سزار، سردار رومی استفاده میشد. در این رمزنگاری حروف الفبا چند قدم به جلو یا عقب کشیده شده و با حروف الفبا دیگر جایگزین میشوند.

**Search insert**  
این الگوریتم یک لیست و عدد را گرفته و در صورت وجود عدد در لیست مکان آن را مشخص میکند اما اگر عدد در لیست وجود نداشته باشد، مشخص میکند که اگر قرار بود عدد وجود داشته باشد در کجا قرار میگرفت.(توجه شود که لیست باید مرتب از کوچک به بزرگ باشد)

**is isomorphic**  
این الگوریتم دو string را گرفته و مشخص میکند که آیا این دو رشته با یکدیگر متقارن هستند یا نه

**a1z26 cipher**  
این الگوریتم هر حرف را با معادل یونیکد عددی خود جایگزین میکند.

فانکشن ord علامت یا حرف مورد نظر را گرفته و یونیکد مربوطه را برمیگرداند و فانکشن chr عدد یونیکد را گرفته و علامت یا حرف مربوطه را نمایش میدهد

ord(‘a’) => 97 chr(98) => b

**bead sort**  
این الگوریتم یک لیست نامرتب را گرفته و آیتم های داخل آن را مرتب میکند