دانشگاه ملی مهارت

آموزشکده میناب

نام و نام خانوادگی : آرش زارعیان

واحد درسی : مباحث ویژه

رشته : مهندسی حرفه ای کامپیوتر

مدرس : محمد احمد زاده

بخش Data Structures and Algorithms 3 :

A. Arrayو List چه تفاوتی دارند؟
B. Dictionaryدر Python چگونه کار می‌کند؟
C. Tupleو List چه تفاوتی دارند؟
D. Setدر Python چرا برای حذف داده‌های تکراری استفاده می‌شود؟
E. Stackو Queue چه تفاوتی دارند؟
F. Hash Tableچیست و چرا کاربرد دارد؟
G. Binary Treeو B-Tree چه تفاوتی دارند؟
H. چرا Graph Data Structure برای شبکه‌های اجتماعی استفاده می‌شود؟
I. Dynamic Programmingچرا در حل مسائل پیچیده کاربرد دارد؟
J. Recursionچیست و چرا در الگوریتم‌های پیشرفته استفاده می‌شود؟

مدرس: محمد احمدزاده
Day... Month... Year... ۲۱ بهمن ۱۴۰۲ Subject... ساحت دیتره

بخش 3 Data structures and Algorithms

A: Array و List چه تفاوتی دارند؟

۱- تعریف کلی

Array: یک ساختار داده‌ای است که شامل مجموعه‌ای از عناصر هم‌نوع با اندازه‌ای ثابت است.

List: یک ساختار داده‌ای پویا است که اندازه آن می‌تواند تغییر کند و معمولاً از اشاره‌گرها برای مدیریت داده‌ها استفاده می‌کند.

۲- اندازه

Array: اندازه آن هنگام تعریف مشخص شده و ثابت است.

List: اندازه آن متغیر است و می‌تواند در زمان اجرا تغییر کند.

۳- عملکرد حافظه

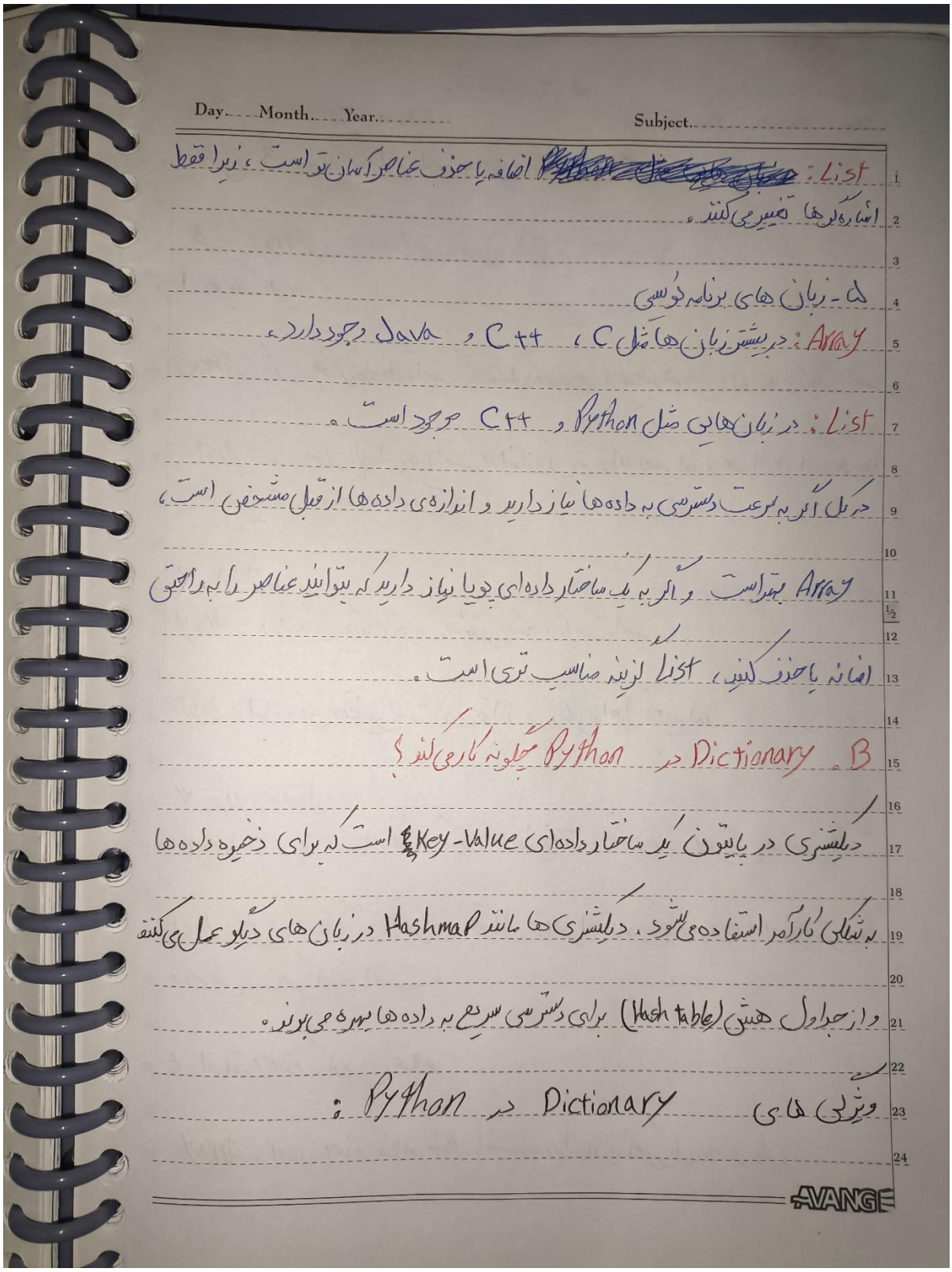
Array: به‌دلیل استفاده از یک بلوک حافظه پیوسته، دسترسی به عناصر آن سریع‌تر است.

List: چون از اشاره‌گرها برای اتصال عناصر استفاده می‌کند، سربار حافظه بیشتری دارد و دسترسی به عناصر کندتر است.

۴- اضافه/حذف کردن عناصر

Array: اضافه یا حذف عناصر مشکل است، زیرا نیاز به تغییر کل ساختار دارد.

AVANGE



Day.... Month.... Year..... Subject.....

List: اضافه یا حذف عناصر آسان تر است، زیرا فقط اشاره گرها تغییر می کنند.

ب- زبان های برنامه نویسی

Array: در بیشتر زبان ها مثل C، C++ و Java وجود دارد.

List: در زبان هایی مثل Python و C++ موجود است.

در کل اگر به سرعت دسترسی به داده ها نیاز دارید و اندازه ی داده ها از قبل مشخص است، Array بهتر است و اگر به یک ساختار داده ای پویا نیاز دارید که بتوانید عناصر را به راحتی اضافه یا حذف کنید، List گزینه مناسب تری است.

B. Dictionary در Python چگونه کار می کند؟

دیکشنری در پایتون یک ساختار داده ای Key-Value است که برای ذخیره داده ها به شکل کارآمد استفاده می شود. دیکشنری ها مانند Hashmap در زبان های دیگر عمل می کنند و از جداول هش (Hash table) برای دسترسی سریع به داده ها بهره می برند.

ویژگی های Dictionary در Python:

Day.... Month.... Year..... Subject.....

۱- هر مقدار با یک کلید مرتبط است ۲- کلیدها باید از نوع تغییرناپذیر باشند

۳- دسترسی سریع ((O(1))

کاربردهای دیکشنری در پایتون:

۱- ذخیره اطلاعات کاربران در یک سیستم ۲- شمارش تعداد کلمات در یک متن

۳- مدیریت داده‌های پایگاه داده در برنامه‌های وب ۴- استفاده در الگوریتم‌های جستجو و هش

C. Tuple و List چه تفاوتی دارند؟

۱- تغییرپذیری

List: قابل تغییر است. می‌توان عناصر آن را تغییر داد، حذف یا اضافه کرد.

Tuple: غیرقابل تغییر است. بعد از ایجاد عناصر آن را نمی‌توان تغییر داد.

۲- سرعت

Tuple: سریع‌تر از List است زیرا تغییرناپذیری باعث بهینه‌تر شدن پردازش و استفاده از حافظه می‌شود.

List: کندتر از Tuple است، به دلیل قابلیت تغییر و اضافه کردن داده‌ها

AVANGE

Day... Month... Year... Subject...

۳ـ مصرف حافظه

Tuple : حافظه کمتری مصرف می‌کند چون ساختار آن ثابت است.

List : به دلیل قابلیت تغییر، حافظه بیشتری به نسبت Tuple اشغال می‌کند.

۴ـ امنیت

Tuple : چون تغییرناپذیر است، برای داده‌های حساس یا ثابت مناسب‌تر است.

List : به دلیل قابلیت تغییر، در برخی مواقع ممکن است ناخواسته تغییر کند.

۵ـ سینتکس

List با [] تعریف می‌شود.

Tuple با () تعریف می‌شود.

D. چرا برای حذف داده‌های تکراری استفاده می‌شود؟ Set در Python

۱ـ عدم پذیرش داده‌های تکراری ۲ـ سرعت بالا در حذف تکراری‌ها

۳ـ سینتکس ساده و راحت

Set بهترین گزینه برای حذف داده‌های تکراری است.

AVANGE

Day... Month... Year... Subject...

E. Stack و Queue چه تفاوتی دارند؟

هردو Stack (پشته) و Queue (صف) از ساختارهای داده‌ای خطی هستند

اما روش دسترسی به داده‌ها در آن متفاوت است.

۱- Stack (پشته)

اصول کار: Lifo (Last in, first out) ← آخرین ورودی، اولین خروجی

مثال: مثل یک دسته بشقاب که آخرین بشقاب گذاشته شده، اولین بشقابی است که برداشته می‌شود.

۲- Queue (صف)

اصول کار: Fifo (first in, first out ← اولین ورودی، اولین خروجی

مثال: مثل صف نانوایی که اولین نفری که وارد شده، اولین نفری است که نان می‌گیرد.

F. Hash Table چیست و چرا کاربرد دارد؟

یک ساختار داده است که برای ذخیره سازی و بازیابی داده‌ها به صورت کارآمد استفاده می‌شود

Hash Table داده‌ها را در قالب Key-Value ذخیره می‌کند به این صورت که هر مقدار از طریق یک کلید قابل دسترسی است.

AVANGE

Day... Month... Year... Subject...

کاربردکن به دلایل زیر است:

۱. سرعت بالا ۲. دسترسی سریع ۳. مدیریت کارآمد داده ها ۴. کاربردهای متنوع

۶. Binary Tree و B-Tree چه تفاوتی دارند؟

Binary Tree (درخت دودویی)

۱. هر گره حداکثر دو فرزند دارد (Left, Right) ۲. معمولا برای پیاده سازی ساختارهایی مثل Binary Search Tree (BST) یا Heap استفاده می‌شود.

B-Tree (درخت B)

۱. یک درخت چند فرزندی (Multi-way Tree) است. ۲. هر گره می‌تواند بیش از ۲ فرزند داشته باشد (معمولا ده‌ها یا صدها فرزند بسته به اندازه بلوک حافظه).

۳. معمولا در سیستم های ذخیره سازی استفاده می‌شود که داده‌ها روی هارد یا SSD هستند.

۴. درخت B همیشه متوازن است.

Day... Month... Year... Subject...

H. چرا Graph Data structure برای شبکه های اجتماعی استفاده می شود؟

شبکه های اجتماعی مثل اینستاگرام، توییتر یا لینکدین همگی بر پایه ارتباطات بین افراد هستند. هر کاربری می تواند با کاربران دیگر ارتباط داشته باشد. این ارتباط ها یک شبکه از افرادی ساخته که به شکل طبیعی یک گراف است.

گره ها (Nodes): کاربران

یال ها (Edges): ارتباط بین کاربران (دوستی، فالو، لایک، کامنت و...)

گراف به دلایل زیر از گراف در شبکه های اجتماعی استفاده می شود:

۱. ارتباطات پیچیده ۲. تحلیل ارتباطات ۳. پیدا کردن کوتاه ترین مسیر

۴. تحلیل خوشه بندی ۵. مدل سازی انعطاف پذیر

I. چرا Dynamic Programming در حل مسائل پیچیده کاربرد دارد؟

برنامه نویسی پویا (DP) یک روش برای حل مسائل است که:

۱. مسئله به زیر مسئله های کوچک تر شکسته می شود. ۲. نتیجه زیر مسئله ها ذخیره می شود

AVANGE

Day. Month. Year. Subject.

تا دوباره محاسبه نشود ۴- با استفاده از نتایج زیر مسئله‌ها، مسئله اصلی حل می‌شود.

به دلایل زیر در مسائل پیچیده کاربرد دارد:

۱- کاهش محاسبات تکراری ۲- بهینه‌سازی زمان ۳- حل بهینه مسائل بهینه‌سازی

۴- حل مسائل دارای هم‌پوشانی ۵- ساختار بهینه‌سازی

۱- Recursion چیست و چرا در الگوریتم‌های پیشرفته استفاده می‌شود؟

Recursion (بازگشت) یک تکنیک در برنامه‌نویسی است که در آن تابع خودش را فراخوانی می‌کند.

یعنی یک مسئله بزرگ را به یک یا چند نسخه کوچک‌تر از همان مسئله تقسیم می‌کنیم و توابع با فراخوانی خودشان این زیرمسئله‌ها را حل می‌کنند و به دلایل زیر در الگوریتم‌های پیشرفته استفاده می‌شود:

۱- تقسیم مسائل پیچیده به مسائل کوچک‌تر ۲- ساختارهای بازگشتی

۳- کاهش پیچیدگی کد ۴- حل مسائل خاص (Backtracking, Divide & Conquer)

معایب: ۱- مصرف زیاد حافظه ۲- خطر stack overflow

۳- در برخی مسائل، بازگشت ساده نیست و حلقه بهتر است