7. Dates and Times

7.1 Represent Dates and Times in MATLAB

MATLAB provides many ways to deal with date and time in form of Date-Time, calendar duration and duration data types. These data types do not only support storing and representing date-times but, also allow operations on date time. The primary way to store date and time information is in datetime arrays, which support arithmetic, sorting, comparisons, plotting, and formatted display. The results of arithmetic differences are returned in duration arrays or, when you use calendar-based functions,in calendarDuration arrays.

DateTime function: The datetime data type specifically represents an instant in time. For example, the datetime function alone returns the current datetime, accurate up to the second.
Syntax: datetime
Output:
ans =

datetime

19-May-2025 22:48:17

Various Options of creating Datetime

1. datetime(relativeDay) uses the date specified by relativeDay. The relativeDay input can be 'today', 'tomorrow', 'yesterday', or 'now'
2. datetime(DateStrings,'InputFormat',infmt) interprets DateStrings using the format specified by infmt.
3. datetime(Y,M,D) creates an array of datetime values for corresponding elements of the Y, M, and D (year, month, day) arrays.
4. datetime(YEAR, MONTH, DAY, HOUR, MINUTE, SECOND) creates an array of datetime values for corresponding elements of the Y, M, D, H,

M, S (year, month, day, hour, minute, second) arrays.

5. datetime(X,'ConvertFrom',dateType) converts the numeric values in X to a datetime array.

6. datetime(DateStrings) creates an array of datetime values from the text in DateStrings representing points in time.

7.2 Customizing Date and Time

1. Create a MATLAB datetime array that represents two dates: June 28, 2014 at 6a.m. and June 28, 2014 at 7 a.m. Specify numeric values for the year, month, day, hour, minute,and second components for the datetime.
   $t = datetime(2014, 6, 28, 6 : 7, 0, 0)$
   Output:
   t =
   28-Jun-2014 06:00:00 28-Jun-2014 07:00:00

2. Change the value of a date or time component by assigning new values to the properties of the datetime array. For example, change the day number of each datetime by assigning new values to the Day property.
   $t.Day = 27 : 28$
   Output:
   t =
   27-Jun-2014 06:00:00 28-Jun-2014 07:00:00

3. Change the display format of the array by changing its Format property. The following format does not display any time components. However, the values in the datetime array do not change.
   $t.Format =' MMMdd, yyyy'$
   Output:
   t =
   Jun 27, 2014 Jun 28, 2014

4. If you subtract one datetime array from another, the result is a duration array in units of fixed length.
   $t2 = datetime(2014, 6, 29, 6, 30, 45)$
   Output:
   t2 =
   29-Jun-2014 06:30:45
   $d = t2 - t$
   Output:
   d =
   48:30:45 23:30:45

5. By default, a duration array displays in the format, hours:minutes:seconds. Change the display format of the duration by changing its Format property. You can display the duration value with a single unit, such as hours.
   $d.Format =' h'$
   Oiutput:
   d =
   48.512 hrs 23.512 hrs

6. You can create a duration in a single unit using the seconds, minutes, hours, days, or years functions. For example, create a duration of 2 days, where each day is exactly 24 hours.
   $d = days(2)$
   Output:
   d =
   2 days

7. Create a datetime array from individual arrays of year, month, and day values.
   $Y = [2014; 2013; 2012];$
   $M = 01;$
   $D = [31; 30; 31];$
   $t = datetime(Y, M, D)$

Output:
t =

3x1 datetime array

31-Jan-2014
30-Jan-2013
31-Jan-2012

8. The input data-time format can be converted to the default format as follows:
$DateStrings = '2014-05-26';'2014-08-03';$
t = datetime(DateStrings,'InputFormat','yyyy-MM-dd')
Output:
t =

2x1 datetime array

26-May-2014
03-Aug-2014

7.3 Calendar Duration

This data type creates arrays of time elapsed in variable calendar units. This data type provides 5 functions for creating arrays of variable units.

- **calmonths** – For creating arrays with the monthly differences in units.
- **calquarters** – For creating arrays with quarterly differences in units.
- **calyears** – For creating arrays with the yearly difference in units.
- **caldays** – For creating arrays with daily differences in units.
- **calweeks** – For creating arrays with the weekly differences in units.

Examples:
(1) calyears(1:2)
Output:
ans =

1x2 calendarDuration array

1y 2y
(2) calquarters(1)
Output:
ans =

calendarDuration

1q
(3) calmonths(1:5)
Output:
ans =

1x5 calendarDuration array

1mo 2mo 3mo 4mo 5mo

Operations on calendar duration arrays

1. You can create a calendar duration in a single unit of variable length. For example, one month can be 28, 29, 30, or 31 days long. Specify a calendar duration of 2 months.
   $L = calmonths(2)$
   Output:
   L =
   2mo
2. Use the caldays, calweeks, calquarters, and calyears functions to specify calendar durations in other units.

Add a number of calendar months and calendar days. The number of days remains separate from the number of months because the number of days in a month is not fixed, and cannot be determined until you add the calendar duration to a specific datetime.
$L = calmonths(2) + caldays(35)$
Output:
L =
2mo 35d

3. Add calendar durations to a datetime to compute a new date.
   $t2 = t + calmonths(2) + caldays(35)$
   Output:
   t2 =
   Oct 01, 2014 Oct 02, 2014
   t2 is also a datetime array.
   whos t2
   Name Size Bytes Class Attributes
   t2 1x2 161 datetime

Duration Display Format

To display a duration as a single number that includes a fractional part (for example, 1.234 hours), specify one of these character vectors:
To specify the number of fractional digits displayed, use the format function.
To display a duration in the form of a digital timer, specify one of the following character vectors.

- 'dd:hh:mm:ss'
- 'hh:mm:ss'
- 'mm:ss'
- 'hh:mm'

Value of Format	Description
'y'	Number of exact fixed-length years. A fixed-length year is equal to 365.2425 days.
'd'	Number of exact fixed-length days. A fixed-length day is equal to 24 hours.
'h'	Number of hours
'm'	Number of minutes
's'	Number of seconds

Operating on Dates and Times

S.No	Problem	Input	Output
1	Create a datetime named GameStart containing the date "August 8, 2015". Create another datetime named GameEnd containing the date "May 17, 2016".	GameStart = datetime(2015,8,8) GameEnd = datetime(2016,5,17)	GameStart = datetime 08-Aug-2015 GameEnd = datetime 17-May-2016
2	Calculate Game length	GameLength = GameEnd - GameStart	GameLength = duration 6792:00:00
3	Convert GameLength to days.	GameLength = days(seasonLength)	GameLength = 283
4	Convert GameLength back to a duration using the days function.	GameLength = days(seasonLength)	GameLength = duration 283 days

Working with months Create a calendar duration named GameLength

from GameStart and GameEnd.
GameLength = between(GameStart,GameEnd)
Output: GameLength =

calendarDuration

9mo 9d

Operating on Vectors of Dates: Upload covid data.

1.	Sort the values in the datetime vector CovidData in ascending order.	CovidData = sort(CovidData)
2.	Calculate the time between the second and first elements of CovidData, and save the output to a duration named t.	t=CovidData(2)- CovidData(1)
3.	Calculate the time between the last and first elements ofmgrHireDate, and save the output to a duration named t2.	t2 = mgrHireDate(end)-mgrHireDate(1)
4.	Calculate how many years the time Difference t corresponds to, saving the result as y. Then find the number of years the time difference t2 corresponds to, and save the result as y2.	y =years(t) y2 =years(t2)

Duration: The duration arrays are similar to calendar duration arrays with the only difference being the length of elapsed time. This data type takes fixed time units and has the following fixed functions of time length:

S.No	Function	Example	Output
1.	years	years(1:5)	ans = 1x5 duration array 1 yr 2 yrs 3 yrs 4 yrs 5 yrs
2.	daya	days(1:2)	ans = 1x2 duration array 1 day 2 days
3.	hours	h = hours(1:5)	h = 1x5 duration array 1 hr 2 hr 3 hr 4 hr 5 hr
4.	minutes	minute(1:5)	ans = 1x4 duration array 1 min 2 min 3 min 4 min
5.	seconds	seconds(0:10)	ans = 1x11 duration array 0 sec 1 sec 2 sec 3 sec 4 sec 5 sec 6 sec 7 sec 8 sec 9 sec 10 sec
6.	milliseconds	milliseconds(1:7)	ans = 1x7 duration array 0.001 sec 0.002 sec 0.003 sec 0.004 sec 0.005 sec 0.006 sec 0.007 sec

Example:dt = datetime(”today”)
dt - h
Output:
ans =

1x5 datetime array

18-May-2025 23:00:00 18-May-2025 22:00:00 18-May-2025 21:00:00 18-May-2025 20:00:00 18-May-2025 19:00:00

7.4 Compare Dates and Time

We can compare dates, times, and durations by using relational operators and comparison functions. Because the datetime and duration data types represent dates and times quantitatively, you can use the same relational operators that you use to compare numeric arrays. However, the comparisons have slightly different meanings, depending on the data type.

- A datetime value can occur before, at the same time as, or after another datetime value.
- A duration value can be shorter than, the same length of time as, or longer than another duration value.
- The calendarDuration data type does not support comparisons using relational operators. Calendar units do not necessarily represent fixed lengths of time.

We can compare two datetime arrays, and you can compare two duration arrays. The arrays must have compatible sizes because relational operators perform element-wise comparisons. In the simplest cases, the two arrays have the same size or one is a scalar. Dates and times can also be represented by text, while durations can also be represented by text and by numbers. Therefore, you can compare datetime arrays to text and duration arrays to text and numbers. Relational operators convert text and numbers to the correct data types before performing operations. You cannot compare a datetime array and a duration array. However, you can compare components of datetime arrays to numbers or to duration arrays.

Examples:

(1) **Compare datetime Values:**

d1 = datetime(”2022-06-05 11:37:05”)

d1 = datetime

05-Jun-2022 11:37:05

d2 = datetime(2022,2:4:10,15,12,0,0)
d2 = 1x3 datetime
15-Feb-2022 12:00:00 15-Jun-2022 12:00:00 15-Oct-2022 12:00:00
Compare the two datetime arrays. The result shows which elements of d2 occur after d1.
$tf = d2 > d1$
tf = 1x3 logical array

0 1 1

(2) **Compare duration Arrays**
Create a duration array.
t1 = duration("03:37:12")
t1 = duration
03:37:12

Create another duration array
t2 = duration(0:2:6,30,0)
t2 = 1x4 duration
00:30:00 02:30:00 04:30:00 06:30:00

Compare the two duration arrays. The result show which elements of t2 are longer than t1.

$tf = t2 > t1$
tf = 1x4 logical array

0 0 1 1

Exercise - 6

1. Specify the current date and time in the local system time zone.
2. Create a datetime named xMin with the value "July 1, 2020". Create another datetime named xMax with the value "Apr 24, 2022".
3. Use numeric inputs to create a datetime named d with a value of "February 22nd , 2021 17:15".
4. Create a 3-by-1 column vector named d2 of type datetime as shown below.
   14-Sep-2017
   14-Sep-2018
   14-Sep-2019
5. Create a 1-by-7 row vector named d3 of type datetime representing the dates October 15th – 21st, 2002.
6. Create a 12-by-1 vector d4 of type datetime containing the 8th day of each month in the year 2016.
7. Subtract xmax from xmin created in problem 3 and save the result to a variable named datdiff.
8. Convert datdiff created above to days. Convert datdiff back to a duration using the days function.
9. Create a duration named startTime that contains 11 hours. Update startTime so that it contains 11 hours and 45 minutes.
10. Create a calendar duration named seasonLength from d1 and d5.
11. Create a calendar duration named "m" containing two calendar months. Save the variable "x" which contains the result that adds "m" to d5.
12. Now what happens if you subtract "m" from d5? Try it and save the result to a variable named y.