

# Date and time

They are ruby class

**These classes have a lot of methods and utilities**

## Date and Time

- Represents a specific point in time
- Time represents the time and the date and has three components
  - Day
  - Month
  - Year
  - Hours
  - Minutes
  - Seconds

- Time.now
- Time.at(linuxtimestamp)
- Others

```
1. Time.now
2. # 2018-10-19 15:43:20 +0200
3.
4. Time.new(2018, 1, 1)
5. # 2018-01-01 00:00:00 +0100
6.
7. Time.at(1500000000)
8. # 2017-07-14 04:40:00 +0200
```

### Linux timestamp

Quantity of time since 1970, it is on milliseconds

add text

```
1. t = Time.now
2.
3. puts t.day
4. puts t.month
5. puts t.hour
```

**Time.local** -> declare time on the current time zone

You can also pass the time-zone as a parameter

## Date and Time

- Time also has an equal operator `<=>`
- You can convert the time to an ASCII string

```
Time.now.asctime    #=> "Wed Apr  9 08:56:03 2003"
Time.now.ctime      #=> "Wed Apr  9 08:56:03 2003"
```

## Date and Time – Daylight Saving Time

- Click to add text

```
Time.local(2000, 1, 1).dst?    #=> false
```

to know if we are on daylight saving time.

## Date and Time - strftime

- Convert a formatted string into a Time

```
- don't pad a numerical output
_ use spaces for padding
0 use zeros for padding
^ upcase the result string
# change case
: use colons for %z
```

**THIS IS A REALLY USEFUL METHOD**

```

1. time = Time.new
2.
3. time.strftime("%d/%m/%Y")      # "05/12/2015"
4. time.strftime("%k:%M")        # "17:48"
5. time.strftime("%I:%M %p")     # "11:04 PM"
6. time.strftime("Today is %A")  # "Today is Sunday"
7. time.strftime("%d of %B, %Y") # "21 of December, 2015"
8. time.strftime("Unix time is %s") # "Unix time is 1449336630"

```

Converts current time to string and it will format it

## Date

# Date



- Date doesn't include the Time component
- Remember to always require it!

```

require 'date'

Date.new(2001,2,3)
#=> #<Date: 2001-02-03 ...>
Date.jd(2451946)
#=> #<Date: 2001-02-03 ...>
Date.ordinal(2001,34)
#=> #<Date: 2001-02-03 ...>
Date.commercial(2001,5,6)
#=> #<Date: 2001-02-03 ...>
Date.parse('2001-02-03')
#=> #<Date: 2001-02-03 ...>
Date.strptime('03-02-2001', '%d-%m-%Y')
#=> #<Date: 2001-02-03 ...>
Time.new(2001,2,3).to_date
#=> #<Date: 2001-02-03 ...>

```

It by default does not include the time part.  
You need to require 'date' to use it.

# Date

- All Date objects are immutable
  - They can't modify themselves
- Accessing specific values with `.year`, `.mon`, `.mday`, `.wday`

```
d = Date.parse('3rd Feb 2001')  
#=> #<Date: 2001-02-03 ...>  
d.year      #=> 2001  
d.mon       #=> 2  
d.mday      #=> 3  
d.wday      #=> 6  
d += 1      #=> #<Date: 2001-02-04 ...>  
d.strftime('%a %d %b %Y') #=> "Sun 04 Feb 2001"
```

# Date

- Available constants in the Date Module

## Constants I

Date::ABBR_DAYNAMES	Array of abbreviated day names
Date::ABBR_MONTHNAMES	Array of abbreviated month names
Date::DAYNAMES	Array of full names of days of the week
Date::ENGLAND	The Julian day number of the day of calendar reform for England and her colonies
Date::GREGORIAN	The Julian day number of the day of calendar reform for the proleptic Gregorian calendar
Date::ITALY	The Julian day number of the day of calendar reform for Italy
Date::JULIAN	The Julian day number of the day of calendar reform for the proleptic Julian calendar
Date::MONTHNAMES	Array of full month names

# Date



- Please note the following operations
  - Why in the third line it shows a 28? Because it takes the last day of the February month.

```
Date.new(2001,3,31) << 2      #=> #<Date: 2001-01-31 ...>
Date.new(2001,3,31) << 1 << 1  #=> #<Date: 2001-01-28 ...>
Date.new(2001,3,31) << 1 << -1  #=> #<Date: 2001-03-28 ...>
```

These operators reduce the months from the date.

**We also go back to the last day of the month** if we put a day greater than the max quantity of days that the month we are trying to get has.

## Active support

### Active Support



- Only available at Rails

```
1. 1.hour.to_i # 3600
2.
3. 1.day        # ActiveSupport::Duration
4. 3.days.ago   #
   ActiveSupport::TimeWithZone
```

With active support we can do these type of things

```
Loading development environment (Rails 7.0.3.1)
3.0.0 :001 > Time.now - 3.days.ago
=> 259199.999967
3.0.0 :002 > Time(Time.now - 3.days.ago)
(irb):2:in `<main>': undefined method `Time' for main:Object (NoMethodError)
3.0.0 :003 > Time.at(Time.now - 3.days.ago)
=> 1970-01-03 17:59:59 34359291691/34359738368 -0600
3.0.0 :004 > █
```

**Homework**

We are gonna use three active support methods related to date and time. Represent that time formatted.