Datetime

Syntax of Running Datetime & Datetime Duration

date(), date.duration()
date.declare()
time()
dt()
dt.design()
date.add(), date.join(), date.pop()
allowed_days()
date.math()

snip.get_smpl(), snip.get_str()

extra

Datetime Table

Syntax of (running) Datetime

%a %ua %va

whole datetime with 24 hours

%k %uk %vk

whole datetime with 12 hours

%b %ub %vb

week**day** (For number, Sunday is 0 then 1 to 6. So, no padding needed.)

%l %xl

month**day**

%B %xB

year**day**

%c %xc

year**week** with **Sunday as first day of week** (At the start of the year, days before first Sunday will be counted as year week 0.)

%m %xm

year**week** with **Monday as first day of week** (At the start of the year, days before first Monday will be counted as year week 0.)

%d %xd %ud %vd

year**month**

%e

year with century

%o %xo

year without century (Padding is 00, 01, ..., 99 while no padding is 0, 1, 2, .., 99)

%f %xf

minute**second** (00, 01, ..., 59)

%p %xp

hour**second** (0000, 0001, ..., 3599)

%F %xF

day**second** (00000, 00001, ..., 86399)

%g %xg

hour**minute** (00, 01, ..., 59)

%q %xq

day**minute** (0000, 0001, ..., 1439)

%h %xh

24 hours

%H %xH

12 hours

%r

prints am, pm

%R

prints AM, PM

%i

microsecond (only padding 000000, ..., 999999)

Suppose variables <datetime>, <date>, <time> have relatable Snips in them.

- **For <datetime>:**

Default design is

> **%l-%d-%e %h:%g:%f**
> 27-08-2019 19:45:17

%a means

> **%l/%d/%e %h:%g:%f**
> 27/08/2019 19:45:17

%k means

> **%l/%d/%e %H:%g:%f %R**
> 27/08/2019 07:45:17 PM

%ua means

> **%ub %ud %l %h:%g:%f %e**
> Saturday August 27 19:45:17 2019

%vk means

> **%vb %vd %l %H:%g:%f %R %e**
> Sat Aug 27 07:45:17 PM 2019

- **For <date>:**

Default design is

> **%l-%d-%e**
> 27-08-2019

%a, %k means

> **%l/%d/%e**
> 27/08/2019

%ua, %uk means

%ub %ud %l %e

Saturday August 27 2019

%va, %vk means

%vb %vd %l %e

Sat Aug 27 2019

- **For <time>:**

Default design, %a, %ua, %va means

%h:%g:%f

19:45:17

%k, %uk, %vk means

%H:%g:%f %R

07:45:17 PM

Note: If %r (or %R) comes after %h (or %xh), it doesn’t print anything. Similarly, if %h (or %xh) comes after %r (or %R), it will be considered as %H (or %xH).

About Datetime duration

%a %ua %va
    whole duration

%b %xb %ub %vb
    days only
%l %xl %ul %vl
    counted days out of total duration

%m %xm %um %vm
    counted weeks out of total duration

%d %xd %ud %vd
    months only
%n %xn %un %vn
    counted months out of total duration

%e %xe %ue %ve
    years only

Suppose variable <duration> has relatable snip in it.

- **For <duration>:**

Default design, %a means
    %b, %d, %e
    26, 07, 18

%ua means
    %ub, %ud, %ue
    26 days, 07 months, 18 years

%va means
    %vb, %vd, %ve
    26d, 07m, 18y

date(), date.duration()

date() generates **date**.
It takes at least two arguments out of three: day, month, year.

<out>{date(27, 8, 2019)} $ ~~27-08-2019~~
<out>{date(none, 8, 2019)} $ ~~08-2019~~
<out>{date(27, none, 2019)} $ ~~27-2019~~
<out>{date(27, 8)} $ ~~27-08~~

Or it takes two arguments: str(), int().

“dd”, 27
“mm”, 8
“yy”, 2019
“d”, 26
“m”, 7
“y”, 18

date.duration() generates **dateDuration**.
It takes at least two arguments out of three: day, month, year.

<out>{date.duration(26, 7, 18)} $ ~~26, 07, 18~~
<out>{date.duration(none, 7, 18)} $ ~~07, 18~~
<out>{date.duration(26, none, 18)} $ ~~26, 18~~
<out>{date.duration(26, 7)} $ ~~26, 07~~

Note: Argument <none> means you don’t want to save that data.

date.declare()

It updates int() to **date**.
For it, it takes one str() as well.

<current_yr>(2019) date.declare(current_yr, “yy”) **is similar to date(“yy”, current_yr : current_yr)**
Other supported str() are: “dd”, “mm”, “d”, “m”, “y”.

time(), dt()

time() generates **time**.
It takes at least two arguments out of three: hour, minute, second.
Or it takes two arguments: str(), int().

```
        “hh”, 19
        “ha”, 7
        “hp”, 7
        “mm”, 45
        “ss”, 17
        “h”, 23
        “m”, 49
        “s”, 49
```

dt() generates **datetime**.
Out of total six arguments, It takes at least one argument out of day, month, year along with at least one out of hour, minute, second.

dt.design(), date.add(), date.join(), date.pop(), allowed_days()

dt.design() updates string data of **date**.
For it, it takes one str() as well.

date.add() adds to/ changes **date**.
For it, it takes one str() and one int().

date.join() adds another date Snip to **date**.

date.pop() pops from **date**.
For it, it takes one str() as well.

        <current>{date(“yy”, 2019)}
        dt.design(current, “%ud %l, %e.”) § ~~2019.~~
        date.add(current, “dd”, 27) **or** date.join(current, date(“dd”, 27)) § ~~27, 2019.~~
        date.add(current, “mm”, 8) § ~~August 27, 2019.~~

```
date.pop(current, “yy”)  § ~~August 27.~~
date.pop(current, “dd”)  § ~~August.~~
date.pop(current, “mm”) **makes it null.base().**

<template>{dt.design(date(none, 1, 2020), “%au”)}  § ~~January 2020~~
<out><Today is ‘date.join(template, date(“dd”, 15))’.>  $ ~~Today is Wednesday January 15 2020.~~


<age>{date.duration(none, 9, 9)}

**Make year 10.**
date.add(age, “y”, 10)

**Add 1 year.**
<updated_yr>(get_smpl(age, “y”)+1) **(of easyA env.)**
date.add(age, “y”, updated_yr)
```

allowed_days() gets **date** (with “mm” must be present. If “mm” is (2), “yy” must be present. “dd” can be present or absent.) and returns int() from 28 to 31.

```
<date>{date(“mm”, 1)}
<day>(26)
if in_range(day, 1, allowed_days(date)):
        date.add(date, “dd”, day)
```

date.math()

It generates **date** or returns null.base().
For it, it takes one str() and two **date**.

For date-date, use “a0” [or “d0”, “m0” or “y0”] **Here, yy-yy = y0, mm-mm = m0, dd-dd = d0. So, it is called “a0” action.**

For date-age, use “a1”
For date+age, use “a3”
For age-age, use “a0” **Here, y-y = y0, m-m = m0, d-d = d0. So, it is called “a0” action.**
For age+age, use “a2”

```
<birth_date>{date(2, 1, 1995)}
<out>{dt.design(date.math(“a0”, date.current(), birth_date), “%va.”}
```

02-01-1995 —> 23-07-2019

02-01-1995 to
02-01-2019
(2019-1995 = 24y)

02-01-2019 to
02-07-2019
(7-1 = 6m)

02-07-2019 to
23-07-2019
(21d)

So, 24y, 6m, 21d

Even if date.math() gets only “d0” action, it does all calculations but returns only “days”.

$ ~~21d, 06m, 24y.~~

snip.get_smpl(), snip.get_str()

get_smpl() gets one Snip and at least one str().
It returns at least one num().

get_str() if gets only Snip, returns string data of it.
Else it gets at least one str() as well and returns at least one relatable string data.

import datetime
from easyA import snip

<date>{dt.design(date(27, 8, 2019), “%du %l, %e”)}

get_smpl(date, “dd”, “mm” : day, month)
<out>{day} $ ~~(27)~~ **Here, null.base() as return (a line generated by outOperator) means Snip doesn’t have that data.**
<out>{month} $ ~~(8)~~

<out>{get_str(date)} $ ~~August 27, 2019~~ **Here, return is never null.base().**
<out><‘get_str(date, “dd”)’ ‘get_str(date, “mm”)’\, ‘get_str(date, “yy”)’>
$ ~~27 August, 2019~~ **Here, null.base() as return (error by outOperator) means Snip doesn’t have (string for) that data.**

dt.design(date, “%au”)
<out>{get_str(date, “dd”)} $ ~~Saturday 27~~ **Here, “dd” gets %bu %l from %au.**

if get_smpl(date, “yy”) = get_smpl(date.current(), “yy”):
    <out><Month is ‘get_str(date, “mm”)’.>
$ ~~Month is August.~~ **prints month only if year is current.**

Extra

date within String Syntax:

import datetime

<detail><I am ‘date(“y”, 28)’ years old.>
Since it doesn’t store Simple data, it is similar to <detail><I am 28 years old.>

To modify <detail> using .replace., first selfArgument must be str().
.replace 28, (date.math(“y2”, date(“y”, 28), date(“m”, 18)))<detail>
§ ~~I am 29 years old.~~

date.declare(get_smpl(date.current(), “yy”), “yy” : current_yr) **(get_smpl() is of easyA env.)**

or <current_yr>{date.pop(date.current(), “dd”)}
date.pop(current_yr, “mm”) § ~~2019~~ **is Snip.**

<current_yr> ++ < is current year.> § ~~2019 is current year.~~ **is now str().**

date as member:

import datetime

<birthdays.list><‘date(31, 12, 1995)’, ‘date(21, 2, 1993)’>
.replace,pos (1), (date.math(“a3”, .get (1){birthdays.list}, date(“d”, 60)))<birthdays.list>
§ ~~<29-02-1996, 21-02-1993>~~ **is fsnip().**

Datetime Table

Alphabet	For running datetime	For datetime duration
a / k	whole	whole duration / -
b / l	day	days only / total counted days
c / m	week	- / total counted weeks
d / n	month	months only / total counted months
e / o	year	years only / -
f / p	second	
g / q	minute	
h / r	hour	
i / s	micro-second	
j / t		
u	full name	
v	short name	
w		
x	no padding	
y		
z		