nse_workday

June 29, 2023

[6]:
```
from nse_workday import get_new_holidays, update_holiday, workday,␣
↪get_holidays_list, get_workdays_list, get_month_weekdays, get_weekdays,␣
↪month_last_weekday, isHoliday
```

1.

get_new_holidays(mode='test')

Retrieves list of latest holidays from nse and save it if fetched list year > last available holidays year in library

Args::

`mode = 'test' / 'normal' . Default 'test' (will not save data.).`

[5]:
```
get_new_holidays(mode='normal')
```

```
No Dates to Exclude
Exceptional Trading Dates (Saturday/Sunday):: ['12-11-2023']
Holidays list :: ['26-01-2023', '07-03-2023', '30-03-2023', '04-04-2023',
'07-04-2023', '14-04-2023', '01-05-2023', '29-06-2023', '15-08-2023',
'19-09-2023', '02-10-2023', '24-10-2023', '14-11-2023', '27-11-2023',
'25-12-2023']
Data already exists
```

2.

update_holiday(dates_set = 'holidays')

Modify / Add / Remove a date from holidays or exceptions list (Exception :: Trading days in Saturday/ Sunday)

Args::

`dates_set : 'holidays' / 'exceptions' . Default 'holidays'`

The date should be entered in dd-mm-yyyy format.

[2]:
```
update_holiday()
```

```
Select an option:
1. Modify a date
2. Add a date
3. Remove a date
```

```
Enter option number: 1
Enter old Date to remove (dd-mm-yyyy): 28-06-2023
Enter the new Date to add (dd-mm-yyyy): 29-06-2023
Do you want to save modified date list?  (Y/N): y
Data saved successfully
```

3.

isHoliday(input_date)

Check the given date is holiday or not . Return Boolean.

Args::

```
input_date : date in %d-%m-%Y/ datetime.date / datetime.datetime format
```

```
[7]: isHoliday(input_date="29-06-2023")
```

```
[7]: True
```

All below functions returns datetime.datetime item / list.

4.

workday(input_date, direction)

Will return the nearest workday in the given direction if the input_date is holiday. If not, will return the same

Args ::

```
input_date : date in %d-%m-%Y / datetime.date / datetime.datetime format

direction : 'next' / 'prev'
```

```
[4]: workday(input_date="29-06-2023",direction='prev')
```

```
[4]: datetime.datetime(2023, 6, 28, 0, 0)
```

5.

get_holidays_list(start_date, end_date)

Returns all holidays as a list within the given range

Args::

```
start_date : date in %d-%m-%Y / datetime.date / datetime.datetime format

end_date : date in %d-%m-%Y / datetime.date / datetime.datetime format
```

```
[10]: get_holidays_list(start_date="20-6-2023", end_date="30-6-2023")
```

```
[10]: [datetime.datetime(2023, 6, 24, 0, 0),
       datetime.datetime(2023, 6, 25, 0, 0),
       datetime.datetime(2023, 6, 29, 0, 0)]
```

6.

get_workdays_list(start_date, end_date)

Returns all workdays as a list within the given range

Args::

```
start_date : date in %d-%m-%Y / datetime.date / datetime.datetime format

end_date : date in %d-%m-%Y / datetime.date / datetime.datetime format
```

[11]:
```
get_workdays_list(start_date="20-6-2023", end_date="30-6-2023")
```

[11]:
```
[datetime.datetime(2023, 6, 20, 0, 0),
 datetime.datetime(2023, 6, 21, 0, 0),
 datetime.datetime(2023, 6, 22, 0, 0),
 datetime.datetime(2023, 6, 23, 0, 0),
 datetime.datetime(2023, 6, 26, 0, 0),
 datetime.datetime(2023, 6, 27, 0, 0),
 datetime.datetime(2023, 6, 28, 0, 0),
 datetime.datetime(2023, 6, 30, 0, 0)]
```

7.

get_month_weekdays(input_date, required_weekday, filtered=True)

Return all occurences of the required weekday from the given date month

Args::

```
input_date : date in %d-%m-%Y / datetime.date / datetime.datetime format

required_weekday : Weekday name (eg. 'Sunday'/'sunday').

filtered : bool . Default True ( Will return the previous day if a required weekday is holiday
```

[12]:
```
get_month_weekdays(input_date="21-6-2023", required_weekday='thursday')
```

[12]:
```
[datetime.datetime(2023, 6, 1, 0, 0),
 datetime.datetime(2023, 6, 8, 0, 0),
 datetime.datetime(2023, 6, 15, 0, 0),
 datetime.datetime(2023, 6, 22, 0, 0),
 datetime.datetime(2023, 6, 28, 0, 0)]
```

8.

month_last_weekday(input_date, last_weekday, filtered=True)

Return last occurence of the required weekday from the given date month

Args::

```
input_date : date in %d-%m-%Y / datetime.date / datetime.datetime format
```

```
last_weekday : Weekday name (eg. 'Sunday'/'sunday').

filtered : bool . Default True (Will return the previous day if a last weekday is holiday.)
```

```
[14]: month_last_weekday(input_date="21-6-2023", last_weekday='thursday')
```

```
[14]: datetime.datetime(2023, 6, 28, 0, 0)
```

9.

get_weekdays(start_date, end_date, required_weekday, filtered=True)

Return all occurences of the required weekday from the given range

Args::

```
start_date : date in %d-%m-%Y / datetime.date / datetime.datetime format

end_date : date in %d-%m-%Y / datetime.date / datetime.datetime format

required_weekday : Weekday name (eg. 'Sunday'/'sunday').

filtered : bool . Default True ( Will return the previous day if a required weekday is holiday
```

```
[15]: get_weekdays(start_date="21-3-2023", end_date="30-6-2023",␣
       ↪required_weekday='friday')
```

```
[15]: [datetime.datetime(2023, 3, 24, 0, 0),
       datetime.datetime(2023, 3, 31, 0, 0),
       datetime.datetime(2023, 4, 6, 0, 0),
       datetime.datetime(2023, 4, 13, 0, 0),
       datetime.datetime(2023, 4, 21, 0, 0),
       datetime.datetime(2023, 4, 28, 0, 0),
       datetime.datetime(2023, 5, 5, 0, 0),
       datetime.datetime(2023, 5, 12, 0, 0),
       datetime.datetime(2023, 5, 19, 0, 0),
       datetime.datetime(2023, 5, 26, 0, 0),
       datetime.datetime(2023, 6, 2, 0, 0),
       datetime.datetime(2023, 6, 9, 0, 0),
       datetime.datetime(2023, 6, 16, 0, 0),
       datetime.datetime(2023, 6, 23, 0, 0),
       datetime.datetime(2023, 6, 30, 0, 0)]
```