CleressianDate Documentation

Lily Ellington

Jan 24, 2021

CONTENTS:

1 CleressianDate 1

2 strftime and strptime Format Tags 7

3 Indices and tables 9

Index 11

CHAPTER

ONE

CLERESSIANDATE

class CleressianDate(*grandCycle=1*, *cycle=1*, *year=1*, *month=1*, *day=1*)

Create a new `CleressianDate` object with the given specification.

Arguments

- **grandCycle** (*int*) – The grand cycle component. Should be an integer.
- **cycle** (*int*) – The cycle component. Must be an integer between 1 and 23.
- **year** (*int*) – The year component. Must be an integer between 1 and 13.
- **month** – The month component. Must be either the full name of a month or an integer between 1 and 10.
- **day** (*int*) – The day component. Must be an integer between 1 and the maximum number of days in the specified month, as returned by `daysInMonth()`.

Throws

- **TypeError** – If any of the arguments are the wrong type.
- **RangeError** – If the additional validations fail.

`CleressianDate.`**static MONTHS**

An array of the month names, as strings. Note that the first element is the empty string to match the 1-indexing of the month numbers.

`CleressianDate.`**get/set grandCycle**([*val*])

The grand cycle component.

Type int

Throws **TypeError** – When setting, if the new value is not an integer.

`CleressianDate.`**get/set cycle**([*val*])

The cycle component.

Type int

Throws

- **TypeError** – When setting, if the new value is not an integer.
- **RangeError** – When setting, if the new value is not between 1 and 23.

`CleressianDate.`**get/set year**([*val*])

The year component.

Type int

Throws

- **TypeError** – When setting, if the new value is not an integer.
- **RangeError** – When setting, if the new value is not between 1 and 13.

`CleressianDate.`**get/set month**([*val*])
The month component.

Type int

Throws

- **TypeError** – When setting, if the new value is neither an integer nor a string.
- **ValueError** – When setting, if the new value is not a valid month name nor an integer between 1 and 10.

`CleressianDate.`**get/set day**([*val*])
The day component.

Type int

Throws

- **TypeError** – When setting, if the new value is not an integer.
- **RangeError** – When setting, if the new value is not between 1 and the number of days in the given month, as returned by `daysInMonth()`.

`CleressianDate.`**get monthName**()
The month name.

Note: This accessor does not have a corresponding setter. Instead, the month name can be set via the *get/set month()* accessor.

Type string

`CleressianDate.`**static fromAbsoluteDate**(*year*[, *day=1*])
An alternate constructor which instead uses absolute year and day, rather than the standard form.

```
>>> let a = CleressianDate.fromAbsoluteDate(677, 43);
>>> a.toString();
"3:7:1 Tiri 9"
```

Arguments

- **year** (*int*) – The absolute year. Note that 1:1:1 = year 1.
- **day=1** (*int*) – The number of the day within the year.

Returns The `CleressianDate` corresponding to the given absolute date.

Return type CleressianDate

Throws

- **TypeError** – If either argument is not an integer.
- **RangeError** – If `day` is not between 1 and 312 (or 313 for a leap year).

`CleressianDate.`**toAbsoluteDate**()
Convert a CleressianDate to an absolute date representation.

```
>>> let a = new CleressianDate(3, 7, 1, 'Tiri', 9);
>>> a.toAbsoluteDate();
Object { year: 677, day: 43 }
```

Returns The absolute date representation. Recall that absolute year 1 is 1:1:1.

Return type Object{year: int, day: int}

`CleressianDate.`**`static isLeapYear`**(*cycle*, *year*)

Determine if _:cycle:year is a leap year.

Arguments

- **`cycle`** (*int*) – The cycle component.
- **`year`** (*int*) – The year component.

Returns `true` if _:cycle:year is a leap year; `false` otherwise.

Return type boolean

`CleressianDate.`**`static daysInMonth`**(*cycle*, *year*, *month*)

Return the number of days in the given month.

Arguments

- **`cycle`** (*int*) – The cycle component.
- **`year`** (*int*) – The year component.
- **`month`** (*int*) – The month component, as an integer.

Returns The number of days in the given month: 34 for all months except Neyu, for which we return 7 in leap years, 6 in non-leap years

Return type int

> **Warning:** Type checks are not performed in this function. As such, if the values passed in are not integers, the function's result may not be accurate. As an example, the function returns `34` whenever `month != 10`.

`CleressianDate.`**`static daysInYear`**(*cycle*, *year*)

Return the number of days in the given year.

Arguments

- **`cycle`** (*int*) – The cycle component.
- **`year`** (*int*) – The year component.

Returns The number of days in _:cycle:year: 313 when this is a leap year, 312 otherwise.

Return type int

> **Warning:** Type checks are not performed in this function. As such, if the values passed in are not integers, the function's result may not be accurate.

`CleressianDate.`**`static distance`**(*x*, *y*)

Return the number of days and years between two dates. The returned values are always positive, regardless of the order of the parameters.

```
>>> let a = new CleressianDate(3, 7, 1, 'Tiri', 9);
>>> let b = new CleressianDate(3, 7, 1, 'Tiri', 11);
>>> CleressianDate.distance(a, b);
Object { years: 0, days: 2 }
>>>
>>> let x = CleressianDate.fromAbsoluteDate(3000, 41);
>>> let y = CleressianDate.fromAbsoluteDate(211, 310);
>>> CleressianDate.distance(x, y);
Obkect { years: 2788, days: 43 }
```

Arguments

- **x, y** (CleressianDate) – the two dates

Returns The number of years and days between the two dates.

Return type Object{years: int, days: int}

CleressianDate.**plus**([*years=0*, *days=0*])

Return a new date with the given number of years and days added.

Arguments

- **years=0** (*int*) – The number of years to add.
- **days=0** (*int*) – The number of days to add.

Returns The date `years` years and `days` days in the future.

Return type CleressianDate

CleressianDate.**minus**([*years=0*, *days=0*])

Return a new date with the given number of years and days subtracted.

Arguments

- **years=0** (*int*) – The number of years to subtract.
- **days=0** (*int*) – The number of days to subtract.

Returns The date `years` years and `days` days in the past.

Return type CleressianDate

CleressianDate.**strftime**(*template*)

Return a string representation according to the template string.

Information about the available format codes can be found *below*.

```
>>> let a = new CleressianDate(3, 7, 1, 'Tiri', 9);
>>> a.strftime('Grand Cycle = %g, Cycle = %c, Year = %y');
'Grand Cycle = 3, Cycle = 7, Year = 1'
```

Arguments

- **template** (*string*) – The template string.

Returns A string representation.

Return type string

Note: This class overrides `.toString` to use this function with the format tag `"%x"`.

`CleressianDate.`**`static strptime`**(*dateStr*[, *template="%x"*])

Returns a `CleressianDate` by parsing the given template string. Any constructor values which are not gvien or inferrable are set to 1.

Information about the available format codes can be found *below*.

```
>>> CleressianDate.strptime('3:7:1 Tiri 9', '%x').toString();
"3:7:1 Tiri 9"
>>>
>>> // partially filled with defaults
>>> CleressianDate.strptime('Tiri 9', '%B %d').toString();
"1:1:1 Tiri 9"
>>>
>>> // competing specification (standard > absolute)
>>> CleressianDate.strptime('0677.043 / 7:16:9 Tiri 6', '%X / %x').toString();
WARN: strptime: grandCycle overwritten: 0677 (3:7:1) -> 1873 (7:7:1)
WARN: strptime: cycle overwritten: 1873 (7:7:1) -> 1990 (7:16:1)
WARN: strptime: year overwritten: 1990 (7:16:1) -> 1998 (7:16:9)
WARN: strptime: day overwritten: .043 (Tiri 09) -> .040 (Tiri 06)
"7:16:9 Tiri 6"
>>>
>>> // competing specification for month (B > b > m)
>>> CleressianDate.strptime("m=2, b=Kle, B=Sirelle", "m=%m, b=%b, B=%B").
↪monthName
WARN: strptime: month overwritten: .035 (Tiri 01) -> .137 (Klesni 01)
WARN: strptime: month overwritten: .137 (Klesni 01) -> .001 (Sirelle 01)
"Sirelle"
>>>
>>> // even when they aren't in that order!
>>> CleressianDate.strptime("B=Sirelle, b=Kle, m=2", "B=%B, b=%b, m=%m").
↪monthName
WARN: strptime: month overwritten: .035 (Tiri 01) -> .137 (Klesni 01)
WARN: strptime: month overwritten: .137 (Klesni 01) -> .001 (Sirelle 01)
"Sirelle"
```

Arguments

- **`dateStr`** (*string*) – The string to parse.
- **`template="%x"`** (*string*) – The format template to use.

Returns The date interpreted through the format string.

Return type CleressianDate

Raises SyntaxError If any tag appears more than once in the template string.

Raises RangeError If the date string does not match the given format template.

Warning: Because of the tag parsing order (`Y` > `j` > `g` > `c` > `y` > `m` > `b` > `B` > `d`), if duplicate values are found (e.g., by passing %Y and %g), values found later in the parsing take priority, regardless of where they occur in the template string. That is, standard form values take priority over absolute form values, and full month name beats abbreviated name beats month index. In instances where these overwrites occur, a warning is issued by `console.warn`.

CleressianDate.**replace**(*repl*)

Return a copy of this date with the given replacements.

```
>>> let a = new CleressianDate(1, 2, 3, 4, 5);
>>> a.replace({grandCycle: 10, day=12}).toString()
'10:2:3 Fis 12'
```

Arguments

- **repl** (*Object*) – An object that contains the replacements to make.

Returns A copy of the date with the given replacements.

Return type CleressianDate

CleressianDate.**copy**()

Return a copy of this date. `a.copy()` is an alias for `a.replace({})`.

Returns An unmodified copy of this date.

Return type CleressianDate

Note: In addition to the above methods:

With the convention that one date is "less than" another when it occurs before the other, the following comparison methods are provided:

Comparison	Provided Method Call
`x == y`	`x.equals(y)` or `x.eq(y)`
`x != y`	`!x.equals(y)`, `!x.eq(y)`, or `x.ne(y)`
`x < y`	`x.lt(y)`
`x <= y`	`x.le(y)`
`x > y`	`x.gt(y)`
`x >= y`	`x.ge(y)`

CHAPTER

TWO

STRFTIME AND STRPTIME FORMAT TAGS

The template strings used in `CleressianDate.strftime()` and `strptime()` may include any of the following format tags:

Tag	CleressianDate Specification	1989 C Specification
%g	number of grand cycle (1, 2, 3, ...)	N/A
%c	number of cycle in grand cycle (1, 2, ..., 23)	Locale's appropriate date/time representation (e.g., en_US: Tue Aug 16 21:30:00 1988)
%y	number of year in cycle (1, 2, ..., 13)	Year without century as a zero-padded decimal number (00, 01, ..., 99)
%b	abbreviated name of month (Sir, Tir, ..., Ney)	Month as locale's abbreviated name (en_US: Jan, Feb, ..., Dec)
%B	full name of month (Sirelle, Tiri, ..., Neyu)	Month as locale's full name (en_US: January, February, ..., December)
%m	number of month (1, 2, ..., 10)	Month as zero-padded decimal number (01, 02, ..., 12)
%d	number of day in month (1, 2, ..., 34)	Day of the month as a zero-padded decimal number (01, 02, ..., 31)
%j	number of day within year (1, 2, ..., 313)	Day of the yar as a zero-padded decimal number (001, 002, ..., 366)
%Y	number of absolute year (1:1:1 = year 1)	Year with century as a decimal number (0001, 0002, ..., 2013, 2014, ..., 9998, 9999)
%%	a literal `%` symbol	A literal `%` symbol

Warning: The C89 specification is provided here for comparison. While some of the tags are the same, a few are completely different—most notably `%c` and `%X`. Even those which are similar, such as `%j`, differ in that the CleressianDate specification does not provide padded values by default.

Instead, padded values can be obtained for the numeric codes by infixing the pad between the percent symbol and the tag key. For example, for a length-four pad for the absolute year, one can write `%04Y`. In the current specifications, only zero-padding is allowed. Formally, the specification requires the following regular expression to be matched: `r'%(0\d+)?K'`, where `K` is the tag key.

Note: The CleressianDate specification also includes `%x` as a shorthand for the standard form representation and `%X` as a shorthand for the absolute form representation. That is,

- `%x` → `%g:%y:%c %B %d`
- `%X` → `%04Y.%03j`

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

C

CleressianDate() (*class*), 1
CleressianDate.static MONTHS (*Cleressian-Date attribute*), 1