numpy.busday_count

`numpy.busday_count(begindates, enddates, weekmask='1111100', holidays=[], busdaycal=None, out=None)`

Counts the number of valid days between *begindates* and *enddates*, not including the day of *enddates*.

If `enddates` specifies a date value that is earlier than the corresponding `begindates` date value, the count will be negative.

New in version 1.7.0.

Parameters:

begindates : *array_like of datetime64[D]*

The array of the first dates for counting.

enddates : *array_like of datetime64[D]*

The array of the end dates for counting, which are excluded from the count themselves.

weekmask : *str or array_like of bool, optional*

A seven-element array indicating which of Monday through Sunday are valid days. May be specified as a length-seven list or array, like [1,1,1,1,1,0,0]; a length-seven string, like '1111100'; or a string like "Mon Tue Wed Thu Fri", made up of 3-character abbreviations for weekdays, optionally separated by white space. Valid abbreviations are: Mon Tue Wed Thu Fri Sat Sun

holidays : *array_like of datetime64[D], optional*

An array of dates to consider as invalid dates. They may be specified in any order, and NaT (not-a-time) dates are ignored. This list is saved in a normalized form that is suited for fast calculations of valid days.

busdaycal : *busdaycalendar, optional*

A `busdaycalendar` object which specifies the valid days. If this parameter is

Skip to main content

If provided, this array is filled with the result.

Returns:

out : *array of int*

An array with a shape from broadcasting `begindates` and `enddates` together, containing the number of valid days between the begin and end dates.

See also

`busdaycalendar`
An object that specifies a custom set of valid days.

`is_busday`
Returns a boolean array indicating valid days.

`busday_offset`
Applies an offset counted in valid days.

Examples

```
>>> # Number of weekdays in January 2011
... np.busday_count('2011-01', '2011-02')
21
>>> # Number of weekdays in 2011
>>> np.busday_count('2011', '2012')
260
>>> # Number of Saturdays in 2011
... np.busday_count('2011', '2012', weekmask='Sat')
53
```

numpy.busday_offset

`numpy.busday_offset(dates, offsets, roll='raise', weekmask='1111100', holidays=None, busdaycal=None, out=None)`

First adjusts the date to fall on a valid day according to the `roll` rule, then applies offsets to the given dates counted in valid days.

New in version 1.7.0.

Parameters:

dates : *array_like of datetime64[D]*

The array of dates to process.

offsets : *array_like of int*

The array of offsets, which is broadcast with `dates`.

roll : *{'raise', 'nat', 'forward', 'following', 'backward', 'preceding', 'modifiedfollowing', 'modifiedpreceding'}, optional*

How to treat dates that do not fall on a valid day. The default is 'raise'.

- 'raise' means to raise an exception for an invalid day.
- 'nat' means to return a NaT (not-a-time) for an invalid day.
- 'forward' and 'following' mean to take the first valid day later in time.
- 'backward' and 'preceding' mean to take the first valid day earlier in time.
- 'modifiedfollowing' means to take the first valid day later in time unless it is across a Month boundary, in which case to take the first valid day earlier in time.
- 'modifiedpreceding' means to take the first valid day earlier in time unless it is across a Month boundary, in which case to take the first valid day later in time.

A seven-element array indicating which of Monday through Sunday are valid days. May be specified as a length-seven list or array, like [1,1,1,1,1,0,0]; a length-seven string, like '1111100'; or a string like "Mon Tue Wed Thu Fri", made up of 3-character abbreviations for weekdays, optionally separated by white space. Valid abbreviations are: Mon Tue Wed Thu Fri Sat Sun

holidays : *array_like of datetime64[D], optional*

An array of dates to consider as invalid dates. They may be specified in any order, and NaT (not-a-time) dates are ignored. This list is saved in a normalized form that is suited for fast calculations of valid days.

busdaycal : *busdaycalendar, optional*

A `busdaycalendar` object which specifies the valid days. If this parameter is provided, neither weekmask nor holidays may be provided.

out : *array of datetime64[D], optional*

If provided, this array is filled with the result.

Returns:

out : *array of datetime64[D]*

An array with a shape from broadcasting `dates` and `offsets` together, containing the dates with offsets applied.

See also

`busdaycalendar`
An object that specifies a custom set of valid days.

`is_busday`
Returns a boolean array indicating valid days.

`busday_count`
Counts how many valid days are in a half-open date range.

Examples

```
>>> # First business day in October 2011 (not accounting for holidays)
... np.busday_offset('2011-10', 0, roll='forward')
numpy.datetime64('2011-10-03')
>>> # Last business day in February 2012 (not accounting for holidays)
... np.busday_offset('2012-03', -1, roll='forward')
numpy.datetime64('2012-02-29')
```

Skip to main content

```
numpy.datetime64('2011-01-19')
>>> # 2012 Mother's Day in Canada and the U.S.
... np.busday_offset('2012-05', 1, roll='forward', weekmask='Sun')
numpy.datetime64('2012-05-13')
```

```
>>> # First business day on or after a date
... np.busday_offset('2011-03-20', 0, roll='forward')
numpy.datetime64('2011-03-21')
>>> np.busday_offset('2011-03-22', 0, roll='forward')
numpy.datetime64('2011-03-22')
>>> # First business day after a date
... np.busday_offset('2011-03-20', 1, roll='backward')
numpy.datetime64('2011-03-21')
>>> np.busday_offset('2011-03-22', 1, roll='backward')
numpy.datetime64('2011-03-23')
```

Previous
numpy.is_busday

Next
numpy.busday_count

numpy.is_busday

`numpy.is_busday(dates, weekmask='1111100', holidays=None, busdaycal=None, out=None)`

Calculates which of the given dates are valid days, and which are not.

New in version 1.7.0.

Parameters:

dates : *array_like of datetime64[D]*

The array of dates to process.

weekmask : *str or array_like of bool, optional*

A seven-element array indicating which of Monday through Sunday are valid days. May be specified as a length-seven list or array, like [1,1,1,1,1,0,0]; a length-seven string, like '1111100'; or a string like "Mon Tue Wed Thu Fri", made up of 3-character abbreviations for weekdays, optionally separated by white space. Valid abbreviations are: Mon Tue Wed Thu Fri Sat Sun

holidays : *array_like of datetime64[D], optional*

An array of dates to consider as invalid dates. They may be specified in any order, and NaT (not-a-time) dates are ignored. This list is saved in a normalized form that is suited for fast calculations of valid days.

busdaycal : *busdaycalendar, optional*

A `busdaycalendar` object which specifies the valid days. If this parameter is provided, neither weekmask nor holidays may be provided.

out : *array of bool, optional*

If provided, this array is filled with the result.

Returns:

out : *array of bool*

An array with the same shape as `dates`, containing True for each valid day

See also

`busdaycalendar`
: An object that specifies a custom set of valid days.

`busday_offset`
: Applies an offset counted in valid days.

`busday_count`
: Counts how many valid days are in a half-open date range.

Examples

```
>>> # The weekdays are Friday, Saturday, and Monday
... np.is_busday(['2011-07-01', '2011-07-02', '2011-07-18'],
...                 holidays=['2011-07-01', '2011-07-04', '2011-07-17'])
array([False, False,  True])
```

Previous
numpy.busdaycalendar

Next
numpy.busday_offset

<u>Пример получения данных из request</u>

```
def add_data(request):
    if request.method == 'GET':
        form = AddData()
    if request.method == 'POST':
        analitic_work = float(request.POST['analitic_work']) / 480
        form = AddData(request.POST)
        obj = form.save()
        obj.analitic_work = float(f"{analitic_work:.2f}")
        obj.save()
        return redirect('success-postdata')
    return render(request, 'osis-dash/engindex/add_data.html', {'form':form})
```

<u>Пример работы NumpyBusday</u>

```
def gen_inf(request):
    today = date.today()
    start_date = str(today.strftime("%Y-%m"))
    end_date_month = int(today.strftime("%m"))
    end_date = str(today.strftime(f"%Y-0{end_date_month+1}"))
    list_holidays = list(Holidays.objects.values_list('holidays_dates',flat=True))
    for holiday_date in list_holidays:
        holidays = holiday_date.split(',')
    busdays = np.busday_count(start_date, end_date, holidays=holidays)

    return render(request, 'osis-dash/engindex/geninfo.html',
    {
      'busdays':busdays
    }
    )
```