Basic Analysis of a Binary Star 💫💫

In this guide, you will learn how to:

- Search for TESS light curves using the Lightkurve package
- Stitch multiple light curves together
- Find the period of a binary star cycle using the light curve
- Use the period to further gain insights on the stars' movement

Packages you will need:

- numpy
- matplotlib
- astropy
- scipy
- lightkurve

1. Import packages

- Make sure you have them downloaded before starting

In [1]:
```
# let's import some packages
# make sure you have them downloaded first!

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

'''
import astropy.units as u
from astropy.table import Table, Column, MaskedColumn, join
from astropy.io import ascii, fits
from astropy.coordinates import SkyCoord
from astropy.io.votable import parse_single_table
'''

plt.style.use('seaborn-notebook')

'''
from scipy.stats import gaussian_kde
from scipy import interpolate
from scipy.optimize import curve_fit
'''

import lightkurve as lk

# from tqdm import tqdm_notebook as tqdm
```

In [2]:
```
###Plot formatting commands
plt.rc('figure', figsize=(11.0, 6.7)) #bigger figures
```

```
plt.rc('axes', titlesize=20)     # fontsize of the axes title
plt.rc('axes', labelsize=20)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=20)    # fontsize of the tick labels
plt.rc('ytick', labelsize=20)    # fontsize of the tick labels
plt.rc('legend', fontsize=18)    # legend fontsize
plt.rc('figure', titlesize=20)  # fontsize of the figure title
# plt.rc('text', usetex=True)    # usetex=True results in an error
font = {'family':'serif','size':20, 'serif':['Times New Roman']}
plt.rc('font',**font)
#plt.style.use('default')
###End formatting commands
```

2. Get the data

- Let's say we've already found a target of interest with coordinates ra = 91.98070326255153, dec = -66.3599094332912.
- We can use these coordinates to search for it using the following command
- Here I specified the author already, but it doesn't need to be specified just yet
- If there are multiple search results, that means multiple light curves were taken at different times, so we stitch them together

In [3]:
```
author = 'QLP' # preprocessing by MIT Quick Look Pipeline
search_result = lk.search_lightcurve('91.98070326255153 -66.3599094332912', auth

search_result
```

Out[3]: SearchResult containing 22 data products.

#	mission	year	author	exptime	target_name	distance
				s		arcsec
0	TESS Sector 01	2018	QLP	1800	41232835	0.0
1	TESS Sector 04	2018	QLP	1800	41232835	0.0
2	TESS Sector 07	2019	QLP	1800	41232835	0.0
3	TESS Sector 10	2019	QLP	1800	41232835	0.0
4	TESS Sector 11	2019	QLP	1800	41232835	0.0
5	TESS Sector 27	2020	QLP	600	41232835	0.0
...	...	...	...	...	...	...
15	TESS Sector 63	2023	QLP	200	41232835	0.0
16	TESS Sector 64	2023	QLP	200	41232835	0.0
17	TESS Sector 65	2023	QLP	200	41232835	0.0
18	TESS Sector 66	2023	QLP	200	41232835	0.0
19	TESS Sector 67	2023	QLP	200	41232835	0.0
20	TESS Sector 68	2023	QLP	200	41232835	0.0
21	TESS Sector 69	2023	QLP	200	41232835	0.0

Length = 22 rows

In [4]:
```
# Note that our search object has multiple entries, representing multiple light
# Let's gather all the data and stitch them together into one giant light curve.

target_name = '41232835'

t_download = ((search_result.author.data == author))

light_curves = search_result[t_download].download_all()

lc = light_curves.stitch(corrector_func=lambda x: x.normalize())
```

```
WARNING: UnitsWarning: 'BJD-2457000, days' did not parse as fits unit: At col 0,
Unit 'BJD' not supported by the FITS standard.  If this is meant to be a custom
unit, define it with 'u.def_unit'. To have it recognized inside a file reader or
other code, enable it with 'u.add_enabled_units'. For details, see https://docs.
astropy.org/en/latest/units/combining_and_defining.html [astropy.units.core]
```

In [5]:
```
# Let's look at the first 5 entries:

lc[:5]
```

Out[5]:

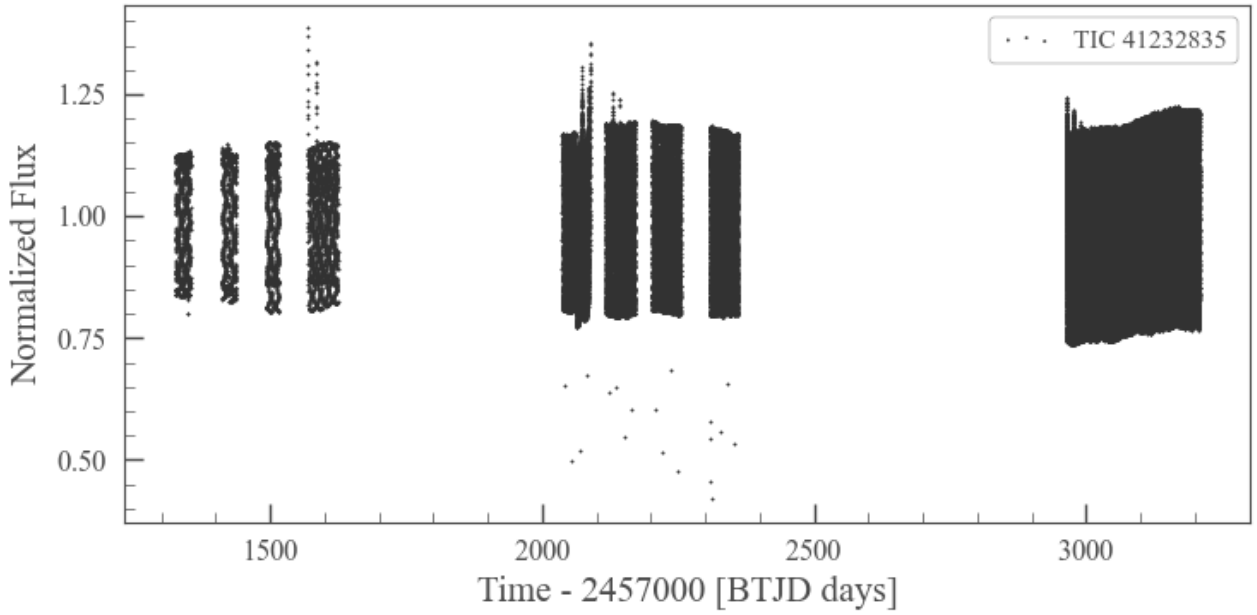
TessLightCurve length=5 LABEL="TIC 41232835" SECTOR=69 AUTHOR=QLP FLUX_ORIGIN=sap_flux

time	flux	flux_err	cadenceno	sap_flux	quality
Time	float32	float64	int32	float32	int32
1325.3234660277578	0.8497300148010254	0.14142780005931854	4697	0.85070276	4096
1325.344299320331	0.9267822504043579	0.14142780005931854	4698	0.9278432	4096
1325.3651326145061	1.0674176216125488	0.14142780005931854	4699	1.0686395	4096
1325.3859659102482	1.12057626247406	0.14142780005931854	4700	1.1218591	4096
1325.4067992075222	1.0453580617904663	0.14142780005931854	4701	1.0465548	4096

In [6]:
```
# this is what the entire lc looks like
lc.scatter()
```

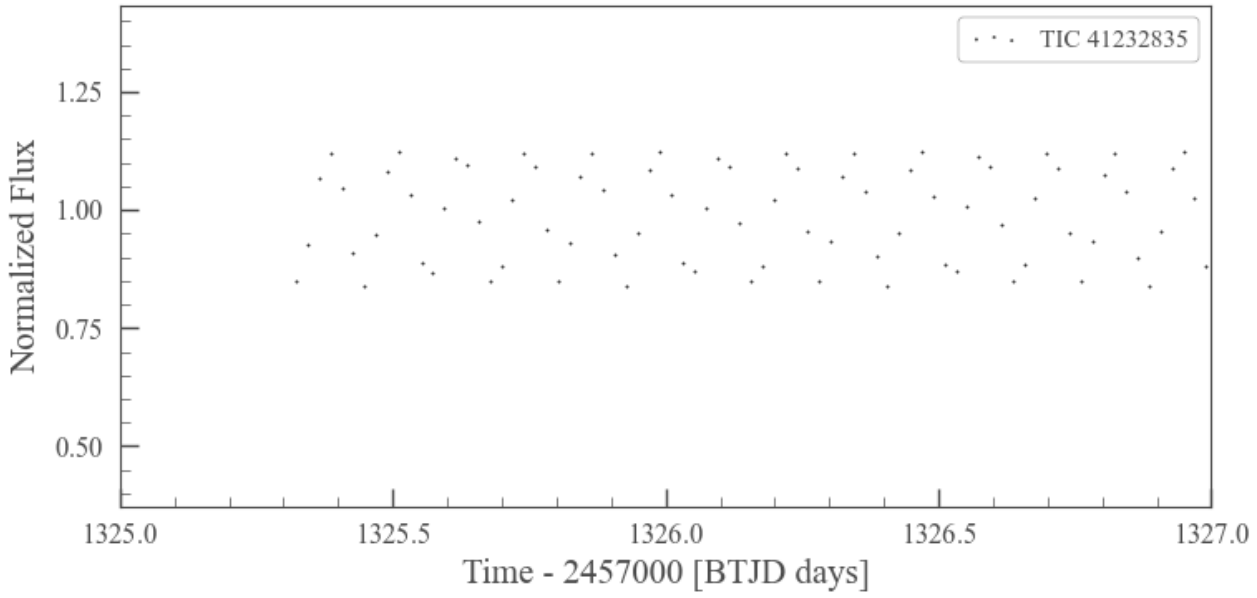
Out[6]:

```
<AxesSubplot:xlabel='Time - 2457000 [BTJD days]', ylabel='Normalized Flux'>
```



In [7]:

```
# we can zoom in
min = 1325
max = 1327

lc.scatter()
plt.xlim(min,max)
plt.show()
```



3. Clean the data

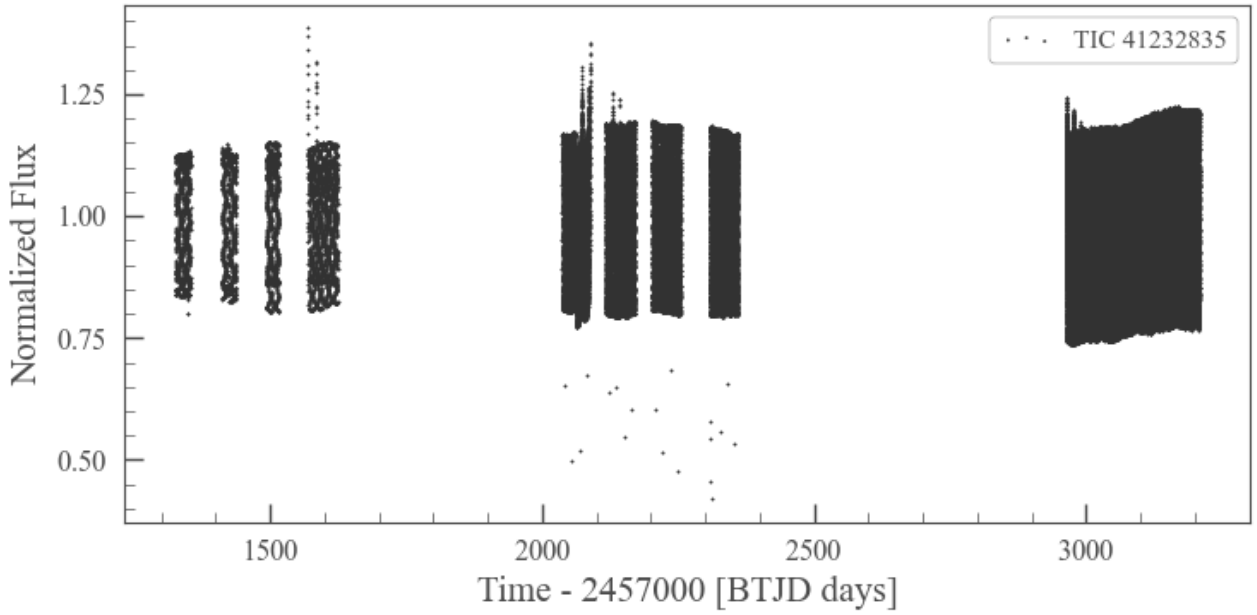
In [8]:

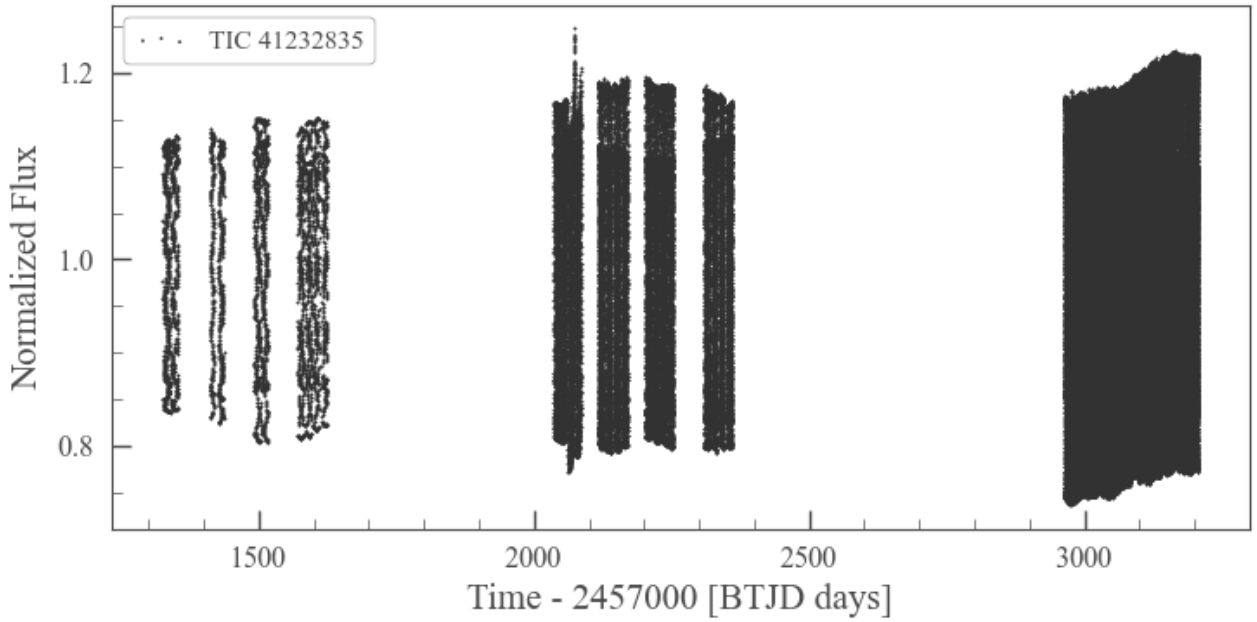
```
# Filter the data so that we reduce some of the noise - put limits on the flux e
# Note that lc is an object in LightCurve object in Lightkurve

lcq = lc[lc['quality']==0]

print("Number of points left:", len(lcq))
print("Number of points eliminated:", len(lc) - len(lcq))
```

```
Number of points left: 120251
Number of points eliminated: 12994
```

In [9]:

```
# compare before and after

lc.scatter()
plt.show()

lcq.scatter()
plt.show()
```





Estimate the orbital period

Orbital period: time taken for both stars to complete one orbit (we measure in days)

Frequency: orbits in a day

The period and frequency are related by the following equation:

$$Frequency = 1/Period$$

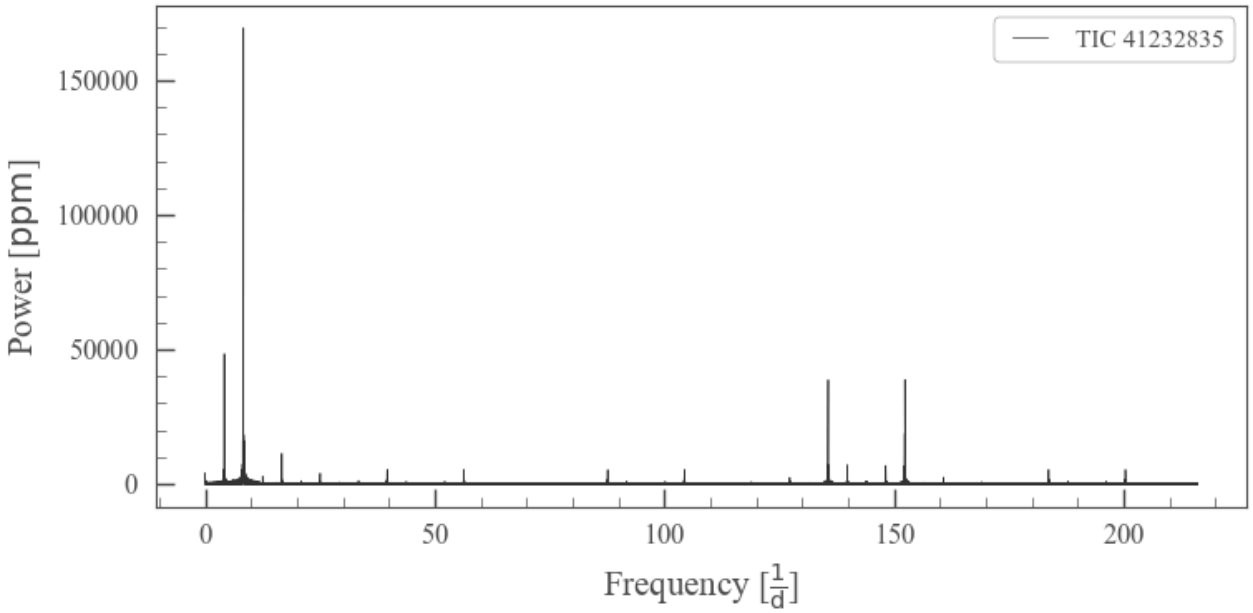
Looking at the zoomed in light curve above, we expect the period will be less than a day. There is a way to determine this more rigorously, and this is by using a periodogram. Essentially, what a periodogram does is similar to a Fourier Transform: it assigns relative strengths to possible period values and returns a graph. Luckily, lightkurve has a handy builtin function to do this!

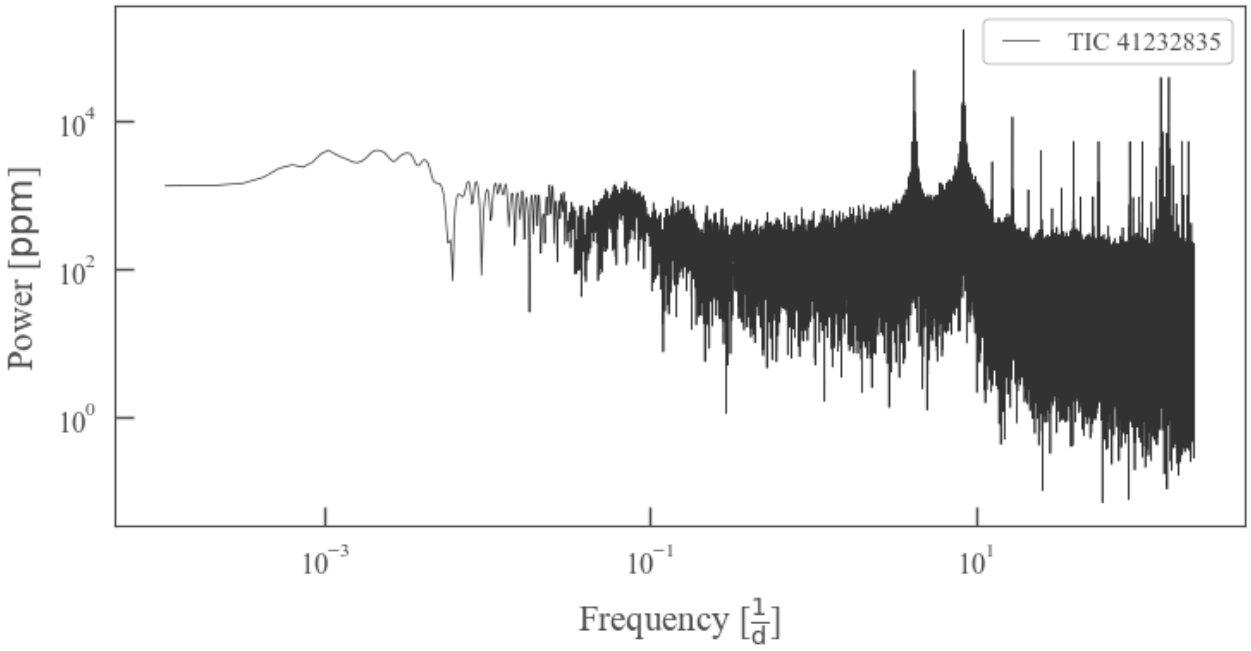
In [10]:
```
pg = lcq.normalize(unit='ppm').to_periodogram()
pg
```

Out[10]:
```
LombScarglePeriodogram(ID: TIC 41232835)
```

In [11]:
```
pg.plot()
pg.plot(scale='log')
```

Out[11]:
```
<AxesSubplot:xlabel='Frequency [$\\mathrm{\\frac{1}{d}}$]', ylabel='Power [$\\ma
thrm{ppm}$]'>
```

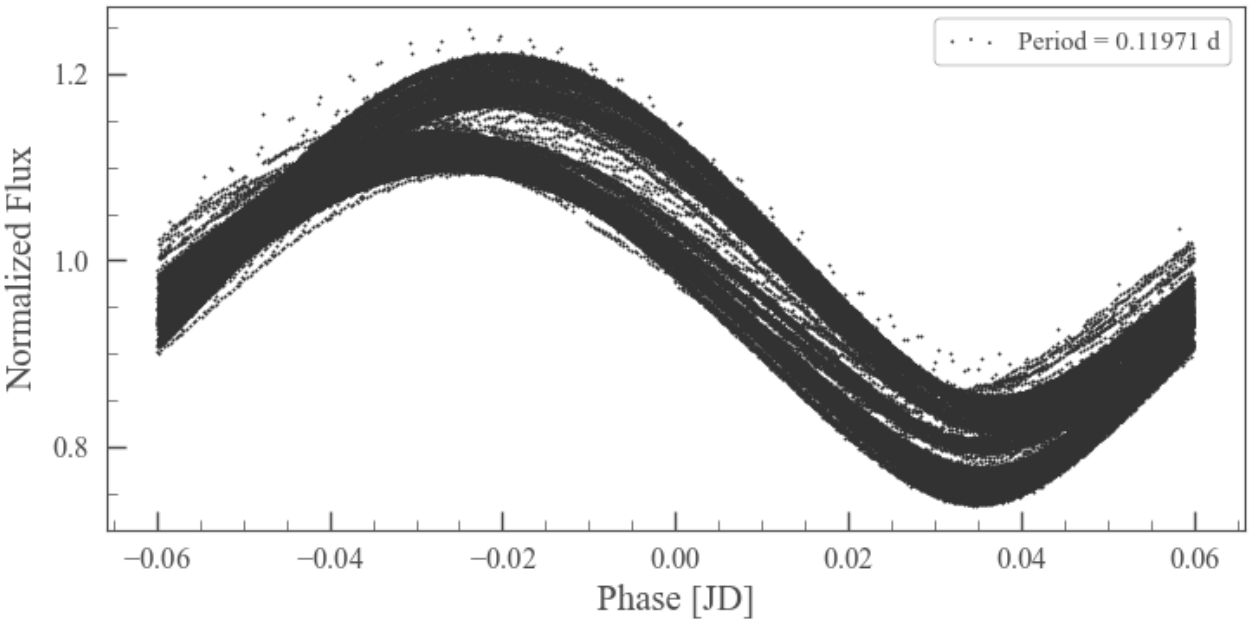




In [12]:
```
period = pg.period_at_max_power
print(period)
lcq.fold(period).scatter(label=f'Period = {period.value:.5f} d')
```

```
0.11970561121532632 d
<AxesSubplot:xlabel='Phase [JD]', ylabel='Normalized Flux'>
```

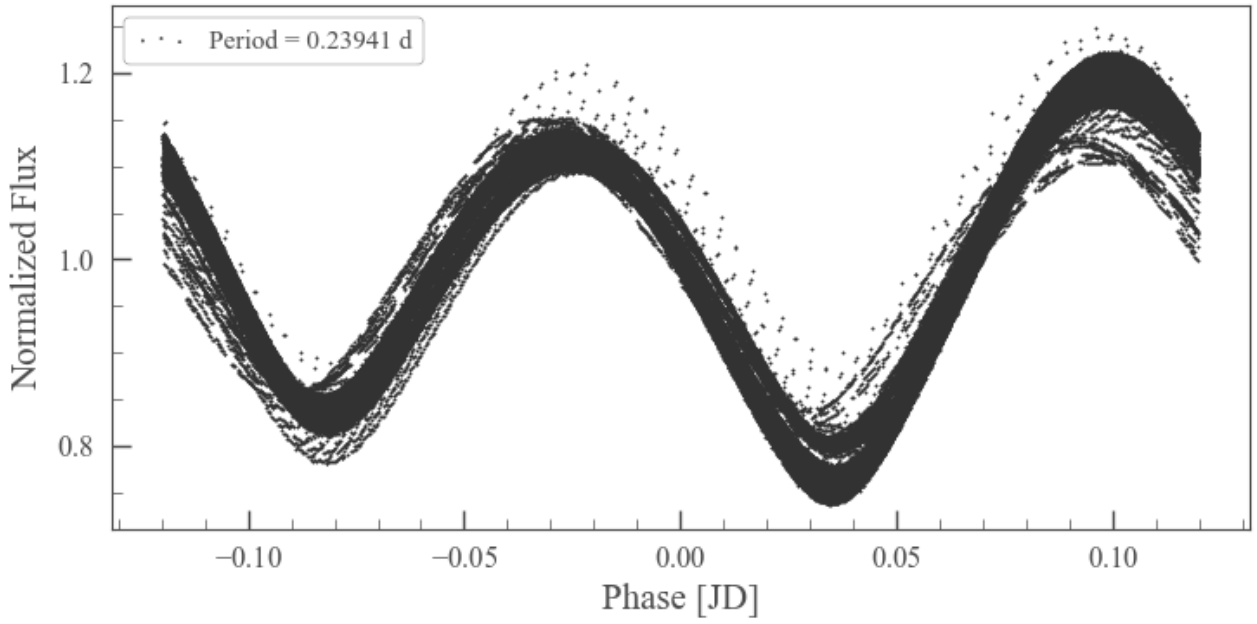
Out[12]:



In [13]:
```
# try multiplying by 2 to account for both eclipses
period *= 2
lcq.fold(period).scatter(label=f'Period = {period.value:.5f} d')
```

```
<AxesSubplot:xlabel='Phase [JD]', ylabel='Normalized Flux'>
```

Out[13]:

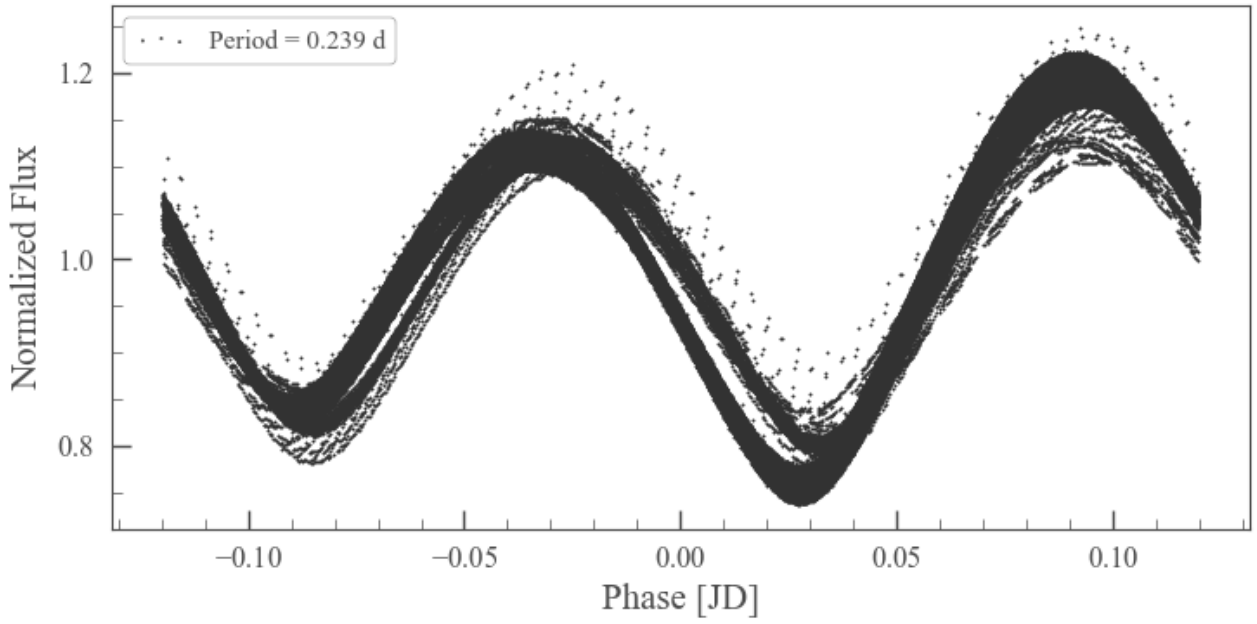


Finding the period of this eclipsing binary is all good and well, but it's not yet very precise, as the fold() plot indicates. Using keyword arguments of the to_periodogram() function, we can improve this estimate. Try one or both of the following:

- Use the minimum_period and maximum_period arguments to limit the range of the periodogram to the area around the peak we are interested in.
- Increase the oversample_factor argument from the default value of 1, which increases the resolution of the periodogram (this is at the expense of correlating the power values, but that is not a problem for the type of analysis we're doing here).

In [14]:
```
# Obtain a better estimate of the period

pg = lcq.to_periodogram(minimum_period = 0.239, maximum_period=0.240, oversample
new_period = pg.period_at_max_power

lcq.fold(new_period).scatter(label=f'Period = {period.value:.3f} d')
new_period
```

Out[14]: $0.23941228\ \mathrm{d}$



In [15]:
```
type(new_period), new_period.value, new_period.unit
```

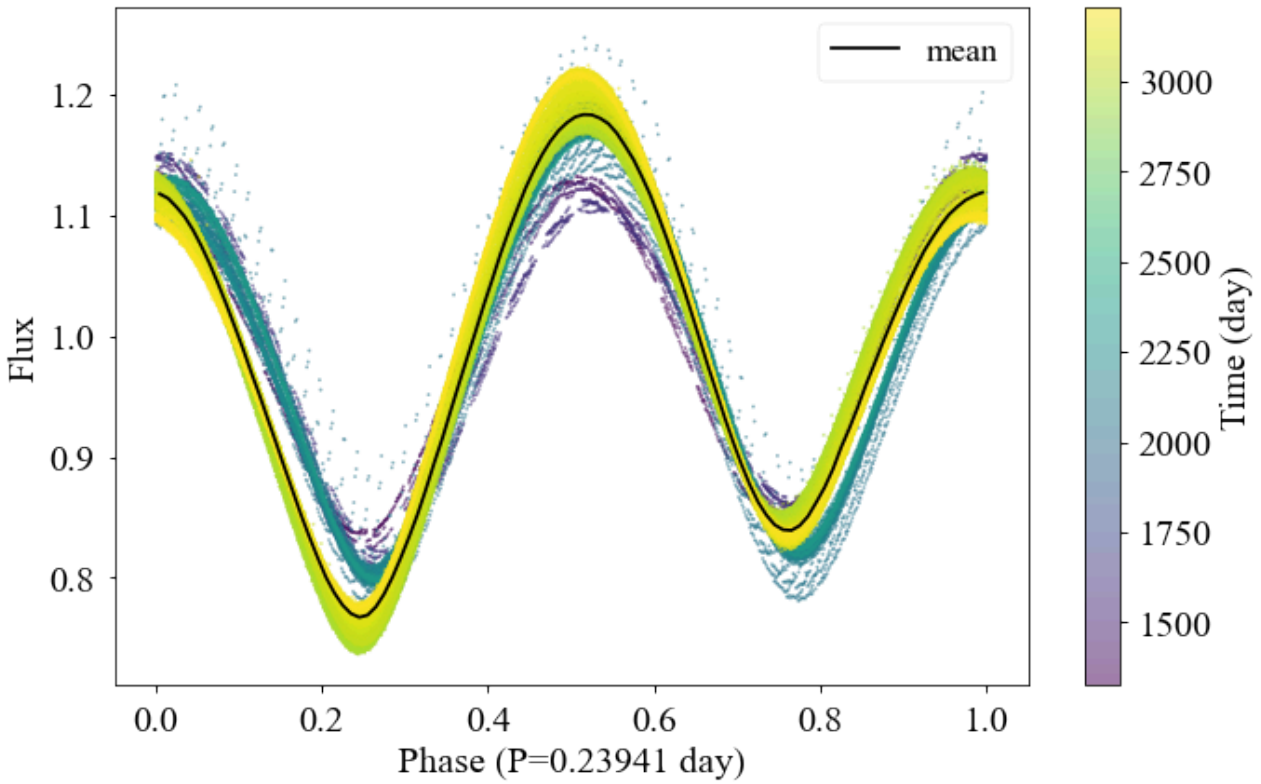
Out[15]:
```
(astropy.units.quantity.Quantity, 0.2394122772192037, Unit("d"))
```

In [16]:
```
def plot_phase_folded_color(lc, period):
    t = (
    lc['quality'] == 0
    )

    P = period

    plt.scatter(
        (lc['time'].value[t]%P)/P,
        lc['flux'].value[t],
        s=2, c=lc['time'].value[t],
        alpha=0.5
    )
    plt.colorbar(label='Time (day)')

    phase_list = np.arange(0., 1.01, 0.01)
    phase_list_center = 0.5 * (phase_list[0:-1] + phase_list[1:])
    mean_y = []
    for ii in range(len(phase_list) - 1):
        tt = (
            t *
            (((lc['time'].value) % P)/P >= phase_list[ii]) *
            (((lc['time'].value) % P)/P < phase_list[ii+1])
        )

        mean_y.append(np.median(lc['flux'][tt].value))

    plt.plot(
        phase_list_center,
        mean_y,
        c='k',
        zorder=10,
        label='mean'
```

```
    )
    plt.xlabel('Phase (P=%.5f day)' %(P))
    plt.ylabel('Flux')
    plt.legend()

    #plt.savefig('figs/20220125_J185444.67+684650.7_lc.png', dpi=200)
    plt.show()
```

In [17]:

```
# Let's make a fancier plot
period = new_period.value
plot_phase_folded_color(lcq, period = period)
```



Just for fun, an animation

In [20]:

```
search_result = lk.search_targetpixelfile('91.98070326255153 -66.3599094332912')
search_result
```

Out[20]: SearchResult containing 34 data products.

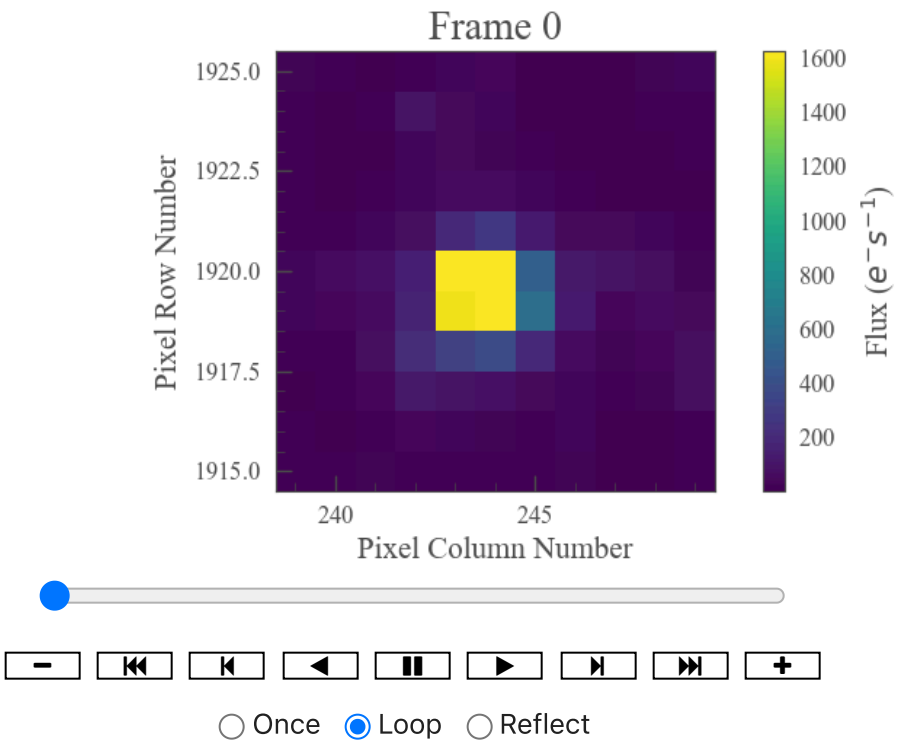
#	mission	year	author	exptime	target_name	distance
				s		arcsec
0	TESS Sector 01	2018	SPOC	120	41232835	0.0
1	TESS Sector 01	2018	TESS-SPOC	1800	41232835	0.0
2	TESS Sector 04	2018	SPOC	120	41232835	0.0
3	TESS Sector 04	2018	TESS-SPOC	1800	41232835	0.0
4	TESS Sector 07	2019	SPOC	120	41232835	0.0
5	TESS Sector 07	2019	TESS-SPOC	1800	41232835	0.0

#	mission	year	author	exptime	target_name	distance
...	...	...	...	...	...	...
27	TESS Sector 65	2023	TESS-SPOC	200	41232835	0.0
28	TESS Sector 66	2023	SPOC	120	41232835	0.0
29	TESS Sector 67	2023	SPOC	120	41232835	0.0
30	TESS Sector 67	2023	TESS-SPOC	200	41232835	0.0
31	TESS Sector 68	2023	SPOC	120	41232835	0.0
32	TESS Sector 68	2023	TESS-SPOC	200	41232835	0.0
33	TESS Sector 69	2023	SPOC	120	41232835	0.0

Length = 34 rows

In [24]:
```
tpf_file = search_result[30].download(quality_bitmask='default')
tpf_file.animate()
```

Out[24]:



In [25]:
```
lk.show_citation_instructions()
```

Out[25]: When using Lightkurve, we kindly request that you cite the following packages:

- lightkurve Copy BibTeX
- astropy Copy BibTeX
- astroquery Copy BibTeX — if you are using *search_lightcurve()* or *search_targetpixelfile()*.
- tesscut Copy BibTeX — if you are using *search_tesscut()*.

In [ ]: