```
from astroquery.mast import Observations
import os
import shutil
import lightkurve as lk
from matplotlib import pyplot as plt
def Bulk_TESS_lc_Query(RA_list, DEC_list, TIC_ID_list, download_dir, host_name_list,
radius = 0.01):
    '''


    Parameters
    ----------
    RA_list : list
        A list of Right Ascensions, in degrees, of your targets.
    DEC_list : list or array-like
        A list of Declinations, in degrees, of your targets.
    TIC_ID_list : list or array-like
        A list of TESS Input Catalog values of your targets.
    download_dir : list or array-like
        The primary directory where the data for each target will be.
    host_name_list : list or array-like
        The names of the target. These will correspond to the folder names
        the data will be assigned to.
    radius : float, optional
        The radius, in degrees, in which to search around the specified RA
        and DEC values. Typically, a value of 0.01 works well.

    Returns
    -------
    Will download TESS .fits file from the SPOC pipeline to the specified
    directory.

    '''
    undownloaded = []
    dirs = []
    for index, TIC in enumerate(RA_list):
        count = 0
        download_dir = download_dir
```

```
    try:
        os.mkdir(download_dir + '/' + str(host_name_list[index]))
    except FileExistsError:
        print('The directory already exists! Skipping the target. If you wish to rerun the data
available for this target, delete or rename the current directory with its name. The run will
continue.')
        dirs.append(download_dir + '/' + str(host_name_list[index]))
        continue
    try:
        obs_table = Observations.query_criteria(s_ra = [(float(RA_list[index]) - radius),
(float(RA_list[index]) + radius)],
                                        s_dec = [(float(DEC_list[index]) - radius),
(float(DEC_list[index]) + radius)],
                                        calib_level = 3,
                                        obs_collection = 'TESS',
                                        dataproduct_type = 'TIMESERIES'
                                        )
        data_products = Observations.get_product_list(obs_table)
    except:
        print('No products with the TIC ID and RA/DEC combination! Try a larger radius. There
may be no data for this target, too.')
        continue

    for indices, items in enumerate(data_products['productFilename']):
        if str(items).endswith('s_lc.fits') == True or str(items).endswith('a_fast-lc.fits') == True:
            if int(items[24:40]) == int(TIC_ID_list[index]):
                try:
                    count += 1
                    print(count)
                    Observations.download_products(data_products[indices], download_dir =
download_dir)
                except:
                    print('There appears to be a server error! This can happen if MAST does not
respond in time. The potentially undownloaded file(s) will appear once the run is finished')
                    undownloaded.append(items)
                    continue
    if count > 0:
        try:
            files = os.listdir(download_dir + '/mastDownload/TESS')
        except:
```

```
            print('There appears to be a directory error! This can happen if your computer does not update its directories fast enough for it to recognize where to put the new file. The potentially undownloaded file(s) will appear once the run is finished')
            undownloaded.append(items)
            continue
        for data in files:
            if str(data).endswith('s') == True:
                try:
                    os.rename(download_dir + '/mastDownload/TESS/' + str(data) + '/' + str(data) + '_lc.fits', download_dir + '/' + str(host_name_list[index]) + '/' + str(data) + '_lc.fits')
                except:
                    print('Warning! Some files may have not downloaded. We skipped it for now, but check at the end for a list of potentially undownloaded files.')
                    undownloaded.append(data)
                    continue
            elif str(data).endswith('fast') == True:
                try:
                    os.rename(download_dir + '/mastDownload/TESS/' + str(data) + '/' + str(data) + '-lc.fits', download_dir + '/' + str(host_name_list[index]) + '/' + str(data) + '-lc.fits')
                except:
                    print('Warning! Some files may have not downloaded. We skipped it for now, but check at the end for a list of potentially undownloaded files.')
                    undownloaded.append(data)
                    continue
        shutil.rmtree(download_dir + '/mastDownload')
    for folders in os.listdir(download_dir):
        if len(os.listdir(download_dir + '/' + folders)) == 0:
            shutil.rmtree(download_dir + '/' + folders)
    print('The already existing directories are:', dirs)
    print('The potential undownloaded files are:', undownloaded)

Bulk_TESS_lc_Query([24.3544618], [-45.6777937], [100100827], '/Users/isaacgutierrez/Downloads', ['WASP-18b'], radius = 0.5)

lc1 = lk.read('/Users/isaacgutierrez/Downloads/WASP-18b/tess2018234235059-s0002-0000000100100827-0121-s_lc.fits')
lc2 = lk.read('/Users/isaacgutierrez/Downloads/WASP-18b/tess2018263035959-s0003-0000000100100827-0123-s_lc.fits')
```

```
lc3 =
lk.read('/Users/isaacgutierrez/Downloads/WASP-18b/tess2020238165205-s0029-000000010010
0827-0193-a_fast-lc.fits')
lc4 =
lk.read('/Users/isaacgutierrez/Downloads/WASP-18b/tess2020238165205-s0029-000000010010
0827-0193-s_lc.fits')
lc5 =
lk.read('/Users/isaacgutierrez/Downloads/WASP-18b/tess2020266004630-s0030-000000010010
0827-0195-a_fast-lc.fits')
lc6 =
lk.read('/Users/isaacgutierrez/Downloads/WASP-18b/tess2020266004630-s0030-000000010010
0827-0195-s_lc.fits')
lc7 =
lk.read('/Users/isaacgutierrez/Downloads/WASP-18b/tess2023237165326-s0069-000000010010
0827-0264-a_fast-lc.fits')
lc8 =
lk.read('/Users/isaacgutierrez/Downloads/WASP-18b/tess2023237165326-s0069-000000010010
0827-0264-s_lc.fits')

plt.figure(figsize=(10, 6))
plt.plot(lc1.time, lc1.flux, label='Original LC 1', alpha=0.7)
```

2)

```
import os
import numpy as np
from astropy.io import fits
import matplotlib.pyplot as plt

# Define directory and list of FITS files
dir_path = "/Users/isaacgutierrez/Desktop/Ampersand/Class Project/WASP-18b"
files = [f for f in os.listdir(dir_path) if f.endswith('.fits')]

flux, time = np.array([]), np.array([])

for lc in files:
    with fits.open(os.path.join(dir_path, lc)) as LC:
        current_flux = LC[1].data['PDCSAP_FLUX']
        current_time = LC[1].data['TIME']
        current_quality = LC[1].data['QUALITY']
```

```
        # Masking bad data
        valid_indices = (~np.isnan(current_flux)) & (np.bitwise_and(current_quality,
0b0101001010111111) == 0)
        masked_flux = current_flux[valid_indices]
        masked_time = current_time[valid_indices]

        if masked_flux.size > 0:  # Check for valid flux data
            masked_flux /= np.median(masked_flux)

        # Concatenate data
        flux = np.concatenate((flux, masked_flux))
        time = np.concatenate((time, masked_time))

# Plotting
plt.scatter(time, flux, marker='.', alpha=0.9)
plt.xlabel('Time (s)')
plt.ylabel('Flux (W/m^2)')
plt.title('Light Curve of WASP-18b (Concatenated)')
plt.show()
```

3)

```
import os
import numpy as np
from astropy.io import fits
import matplotlib.pyplot as plt
import astropy.timeseries as at

# Step 1: Reload the light curve
dir_path = "/Users/isaacgutierrez/Desktop/Ampersand/Class Project/WASP-18b"
files = [f for f in os.listdir(dir_path) if f.endswith('.fits')]
flux, time = np.array([]), np.array([])

for lc in files:
    with fits.open(os.path.join(dir_path, lc)) as LC:
        current_flux = LC[1].data['PDCSAP_FLUX']
        current_time = LC[1].data['TIME']
        current_quality = LC[1].data['QUALITY']

        # Masking bad data
```

```
        valid_indices = (~np.isnan(current_flux)) & (np.bitwise_and(current_quality,
0b0101001010111111) == 0)
        masked_flux = current_flux[valid_indices]
        masked_time = current_time[valid_indices]

        if masked_flux.size > 0:  # Check for valid flux data
            masked_flux /= np.median(masked_flux)

        # Concatenate data
        flux = np.concatenate((flux, masked_flux))
        time = np.concatenate((time, masked_time))

# Step 2: Plot the light curve
plt.figure(figsize=(10, 6))
plt.scatter(time, flux, marker='.', alpha=0.7)
plt.xlabel('Time (days)')
plt.ylabel('Normalized Flux')
plt.title('Light Curve of WASP-18b (Concatenated)')
plt.show()

# Step 3: Use BLS to find periodic transits
# Define the time and flux for BLS
time -= np.min(time)  # Normalize time
flux_err = np.std(flux)  # Estimate flux error as standard deviation

# Initialize the BLS model
bls = at.BoxLeastSquares(time, flux)

# Define the period range for search
periods = np.linspace(0.5, 10, 10000)
bls_result = bls.power(periods, duration=0.1)

# Step 4: Plot the periodogram
plt.figure(figsize=(10, 6))
plt.plot(bls_result.period, bls_result.power, color='b')
plt.xlabel('Period (days)')
plt.ylabel('Power (W)')
plt.title('BLS Periodogram of WASP-18b Light Curve (Concatenated)')
plt.show()
```

```
# Step 5: Identify the strongest period
best_period = bls_result.period[np.argmax(bls_result.power)]
print(f'(Best-Fit) Period: {best_period} days')

# Step 6: Fold the light curve at the best period
folded_time = (time % best_period) / best_period  # Phase
sorted_indices = np.argsort(folded_time)
folded_time = folded_time[sorted_indices]
folded_flux = flux[sorted_indices]

# Step 7: Calculate the depth of the transit
# Bin the data to reduce noise and isolate the transit
bin_width = 0.01  # Phase width for binning
bins = np.arange(0, 1 + bin_width, bin_width)
binned_flux, bin_edges = np.histogram(folded_time, bins=bins, weights=folded_flux)
binned_counts, _ = np.histogram(folded_time, bins=bins)
binned_flux /= binned_counts  # Average flux per bin

# Find the minimum flux in the binned light curve
transit_depth = 1 - np.min(binned_flux)
print(f"Transit Depth: {transit_depth:.6f}")

# Step 8: Calculate the planet radius
Rs = 1.319  # Stellar radius in solar radii
Rp = Rs * (transit_depth ** 0.5)
print(f"Radius of Planet: {Rp:.2f} Solar Radii")

a = 0.02024*215.032
i = 1.45735
b = np.cos(i)*(a/Rs)
print(f'Impact Parameter: {b}')

#Explain math + physics in Latex doc.
#Desired results:
   #Period: 0.94 days
   #Radius: 0.11 Solar radii
   #Impact Parameter: ~0.37
```

4)

```
import os
```

```
import numpy as np
from astropy.io import fits
import matplotlib.pyplot as plt
import astropy.timeseries as at
import lightkurve as lk

# Step 1: Reload the light curve
dir_path = "/Users/isaacgutierrez/Desktop/Ampersand/Class Project/WASP-18b"
files = [f for f in os.listdir(dir_path) if f.endswith('.fits')]
flux, time = np.array([]), np.array([])

for lc in files:
    with fits.open(os.path.join(dir_path, lc)) as LC:
        current_flux = LC[1].data['PDCSAP_FLUX']
        current_time = LC[1].data['TIME']
        current_quality = LC[1].data['QUALITY']

        # Masking bad data
        valid_indices = (~np.isnan(current_flux)) & (np.bitwise_and(current_quality,
0b0101001010111111) == 0)
        masked_flux = current_flux[valid_indices]
        masked_time = current_time[valid_indices]

        if masked_flux.size > 0:  # Check for valid flux data
            masked_flux /= np.median(masked_flux)

        # Concatenate data
        flux = np.concatenate((flux, masked_flux))
        time = np.concatenate((time, masked_time))

# Step 2: Plot the light curve
plt.figure(figsize=(10, 6))
plt.scatter(time, flux, marker='.', alpha=0.5)
plt.xlabel('Time (days)')
plt.ylabel('Normalized Flux (W/mˆ2)')
plt.title('Light Curve of WASP-18b (Concatenated)')
plt.show()

# Step 3: Use BLS to find periodic transits
# Define the time and flux for BLS
```

```
time -= np.min(time)  # Normalize time
flux_err = np.std(flux)  # Estimate flux error as standard deviation

# Initialize the BLS model
bls = at.BoxLeastSquares(time, flux)

# Define the period range for search
periods = np.linspace(0.8, 1.2, 10000)

# Adjust the duration for each period to avoid the error
# Transit duration is 5% of period, ensure it's always smaller than the period
durations = np.minimum(periods * 0.05, periods * 0.9)

bls_result = bls.power(periods, duration=durations)  # Use dynamic durations

# Step 4: Plot the periodogram
plt.figure(figsize=(10, 6))
plt.plot(bls_result.period, bls_result.power, color='b')
plt.xlabel('Period (days)')
plt.ylabel('Power (W)')
plt.title('BLS Periodogram of WASP-18b Light Curve (Concatenated)')
plt.show()

# Step 5: Identify the strongest period
model = at.BoxLeastSquares(time, flux)
results = model.power(periods, durations)
index = np.argmax(results.power)

# Desired results:
# Period: 0.94 days
# Radius: 1.24 Jupiter radii
# Impact Parameter: ~0.32

best_period = bls_result.period[np.argmax(bls_result.power)]
print(f"Best-fit Period: {best_period:.4f} days")
t0 = results.transit_time[index]
duration = results.duration[index]

lc = lk.LightCurve(time=masked_time, flux=masked_flux)
folded_lc = lc.fold(period=best_period, epoch_time=t0)
```

```
# Shift the phase for three separate transits
phase = folded_lc.time.value
flux = folded_lc.flux.value

# Shift phases for three distinct transits
phase_shifted_pos1 = phase + best_period  # First shift
phase_shifted_pos2 = phase + 2 * best_period  # Second shift

# Concatenate the phases for three different transits
all_phases = np.concatenate((phase, phase_shifted_pos1, phase_shifted_pos2))
all_fluxes = np.concatenate((flux, flux, flux))

# Plot the three distinct transits
plt.scatter(all_phases, all_fluxes, s=5)
plt.xlabel('Phase (Radians)')
plt.ylabel('Normalized Flux (W/mˆ2)')
plt.title('Three Separate Transits of WASP-18b')
plt.xlim(-1.5 * best_period, 3 * best_period)  # Adjust x-axis to fit three transits
plt.show()

# Scale the phase to stretch transits horizontally
scaled_phase = all_phases * 2.5 # Adjust the scale factor as needed

# Plot the horizontally stretched transits
plt.figure(figsize=(10, 6))
plt.scatter(scaled_phase, all_fluxes, s=10, alpha=0.5, label="Raw Data")

#Apply binning for clarity
bins = np.linspace(scaled_phase.min(), scaled_phase.max(), 500)  # Define bins
bin_indices = np.digitize(scaled_phase, bins)  # Assign phases to bins
binned_flux = [np.mean(all_fluxes[bin_indices == i]) for i in range(len(bins))]
plt.plot(bins, binned_flux, color='r', linewidth=1.5, label="Binned Data")

# Label and format the plot
plt.xlabel('Scaled Phase (Radians)')
plt.ylabel('Normalized Flux (W/mˆ2)')
plt.title('(Horizontally Stretched) Transits of WASP-18b')
plt.xlim(-1.5 * best_period * 1, 3 * best_period * 2.25)  # Adjust x-axis range based on scaling
plt.legend()
```

```
plt.show()
```

5)

```
import os
import numpy as np
from astropy.io import fits
import matplotlib.pyplot as plt
import astropy.timeseries as at
import lightkurve as lk

# Step 1: Reload the light curve
dir_path = "/Users/isaacgutierrez/Desktop/Ampersand/Class Project/WASP-18b"
files = [f for f in os.listdir(dir_path) if f.endswith('.fits')]
flux, time = np.array([]), np.array([])

for lc in files:
    with fits.open(os.path.join(dir_path, lc)) as LC:
        current_flux = LC[1].data['PDCSAP_FLUX']
        current_time = LC[1].data['TIME']
        current_quality = LC[1].data['QUALITY']

        # Masking bad data
        valid_indices = (~np.isnan(current_flux)) & (np.bitwise_and(current_quality,
0b0101001010111111) == 0)
        masked_flux = current_flux[valid_indices]
        masked_time = current_time[valid_indices]

        if masked_flux.size > 0:  # Check for valid flux data
            masked_flux /= np.median(masked_flux)

        # Concatenate data
        flux = np.concatenate((flux, masked_flux))
        time = np.concatenate((time, masked_time))

# Step 2: Plot the light curve
plt.figure(figsize=(10, 6))
plt.scatter(time, flux, marker='.', alpha=0.5)
plt.xlabel('Time (days)')
plt.ylabel('Normalized Flux')
plt.title('Light Curve of WASP-18b (Concatenated)')
```

```
plt.show()

# Step 3: Use BLS to find periodic transits
# Define the time and flux for BLS
time -= np.min(time)  # Normalize time
flux_err = np.std(flux)  # Estimate flux error as standard deviation

# Initialize the BLS model
bls = at.BoxLeastSquares(time, flux)

# Define the period range for search
periods = np.linspace(0.8, 1.2, 10000)

# Adjust the duration for each period to avoid the error
# Transit duration is 5% of period, ensure it's always smaller than the period
durations = np.minimum(periods * 0.05, periods * 0.9)

bls_result = bls.power(periods, duration=durations)  # Use dynamic durations

# Step 4: Plot the periodogram
plt.figure(figsize=(10, 6))
plt.plot(bls_result.period, bls_result.power, color='b')
plt.xlabel('Period (days)')
plt.ylabel('Power')
plt.title('BLS Periodogram')
plt.show()

# Step 5: Identify the strongest period
model = at.BoxLeastSquares(time, flux)
results = model.power(periods, durations)
index = np.argmax(results.power)

# Get the best period, transit time, and duration
best_period = bls_result.period[np.argmax(bls_result.power)]
print(f"Best-fit Period: {best_period:.4f} days")
t0 = results.transit_time[index]
duration = results.duration[index]

lc = lk.LightCurve(time=masked_time, flux=masked_flux)
folded_lc = lc.fold(period=best_period, epoch_time=t0)
```

```
# Shift the phase for three separate transits
phase = folded_lc.time.value
flux = folded_lc.flux.value

# Step 1: Identify the regions of the two primary transits and mask them
# The first primary transit is around phase 0
# The second primary transit is around phase best_period, so around phase ~ 0.94
mask_primary_transits = ((phase > -0.05) & (phase < 0.05)) | ((phase > best_period - 0.05) &
(phase < best_period + 0.05))

# Step 2: Exclude the flux values corresponding to the primary transits
phase_no_primary = phase[~mask_primary_transits]
flux_no_primary = flux[~mask_primary_transits]

# Step 3: Plot only the secondary transits (after excluding the two primary transits)
plt.scatter(phase_no_primary, flux_no_primary, s=5, color='orange', alpha=0.7)
plt.xlabel('Phase')
plt.ylabel('Normalized Flux')
plt.title('Secondary Transits of WASP-18b (without primary transits)')
plt.xlim(-1.5 * best_period, 3 * best_period)  # Adjust x-axis to fit the remaining transits
plt.show()

# Step 4: Optionally, adjust the scaling for the secondary transits
scaled_phase_no_primary = phase_no_primary * 4  # Adjust the scale factor as needed

# Plot the horizontally stretched secondary transits
plt.figure(figsize=(10, 6))
plt.scatter(scaled_phase_no_primary, flux_no_primary, s=10, alpha=0.5, label="Raw Data
(Secondary Transits)")

# Optional: Apply binning for clarity
bins = np.linspace(scaled_phase_no_primary.min(), scaled_phase_no_primary.max(), 500)  #
Define bins
bin_indices = np.digitize(scaled_phase_no_primary, bins)  # Assign phases to bins
binned_flux = [np.mean(flux_no_primary[bin_indices == i]) for i in range(len(bins))]
plt.plot(bins, binned_flux, color='r', linewidth=1.5, label="Binned Data")

# Label and format the plot
plt.xlabel('Scaled Phase (Radians)')
```

```
plt.ylabel('Normalized Flux (W/mˆ2)')
plt.title('Horizontally Stretched Secondary Transits of WASP-18b')
plt.xlim(-1.5 * best_period * 1, 2)  # Adjust x-axis range based on scaling
plt.legend()
plt.show()

# Step 1: Identify the flux values during the secondary transit
# We've already excluded the primary transits and are now working with secondary transits

# We assume the secondary transit flux corresponds to flux values around phase ~0.5, phase ~1.5, etc.
# Let's define the phase windows for the secondary transits. We expect the secondary transit around phase ~0.5.
# Adjust based on your results, if needed.

# Define phase window for secondary transit (e.g., phase ~0.5 ± 0.05)
mask_secondary_transits = ((phase_no_primary > 0.45) & (phase_no_primary < 0.55)) | \
                          ((phase_no_primary > 1.45) & (phase_no_primary < 1.55))

# Extract flux values during the secondary transit
secondary_transit_flux = flux_no_primary[mask_secondary_transits]

# Step 2: Calculate the out-of-transit baseline flux
# Baseline flux is calculated from the data that doesn't fall within the secondary transit phase windows
mask_out_of_transit = ~mask_secondary_transits
out_of_transit_flux = flux_no_primary[mask_out_of_transit]

# Step 3: Calculate the baseline flux (average of the out-of-transit flux)
baseline_flux = np.mean(out_of_transit_flux)

# Step 4: Calculate the mean flux during the secondary transit
secondary_transit_mean_flux = np.mean(secondary_transit_flux)

# Step 5: Calculate the secondary transit depth
secondary_depth = 1 - (secondary_transit_mean_flux / baseline_flux)

# Output the secondary transit depth
print(f"Secondary Transit Depth: {secondary_depth:.4f}")
```

```
# Given values
R_sun = 6.96e8  # Solar radius in meters
R_jupiter = 7.1492e7  # Jupiter radius in meters
R_star = 1.378 * R_sun  # Star radius in meters (WASP-18)
R_planet = 1.2 * R_jupiter  # Planet radius in meters (WASP-18b)
T_star = 6400  # Star temperature in K (WASP-18)
delta_F = 0.0011  # Secondary eclipse depth

# Calculate the dayside temperature of the planet
T_planet = T_star * (R_star / R_planet) * (delta_F)**(6/13)
print(f'Temperature of WASP-18b (Day-Side) (K): {T_planet}')
```

6)

```
# Just edit this to show primary and secondary transits. Explain phase modulations (whatever
that means)
import os
import numpy as np
from astropy.io import fits
import matplotlib.pyplot as plt
import astropy.timeseries as at
import lightkurve as lk

# Step 1: Reload the light curve
dir_path = "/Users/isaacgutierrez/Desktop/Ampersand/Class Project/WASP-18b"
files = [f for f in os.listdir(dir_path) if f.endswith('.fits')]
flux, time = np.array([]), np.array([])

for lc in files:
    with fits.open(os.path.join(dir_path, lc)) as LC:
        current_flux = LC[1].data['PDCSAP_FLUX']
        current_time = LC[1].data['TIME']
        current_quality = LC[1].data['QUALITY']

        # Masking bad data
        valid_indices = (~np.isnan(current_flux)) & (np.bitwise_and(current_quality,
0b0101001010111111) == 0)
        masked_flux = current_flux[valid_indices]
        masked_time = current_time[valid_indices]
```

```
        if masked_flux.size > 0:  # Check for valid flux data
            masked_flux /= np.median(masked_flux)

        # Concatenate data
        flux = np.concatenate((flux, masked_flux))
        time = np.concatenate((time, masked_time))

# Step 2: Plot the light curve
plt.figure(figsize=(10, 6))
plt.scatter(time, flux, marker='.', alpha=0.5)
plt.xlabel('Time (days)')
plt.ylabel('Normalized Flux')
plt.title('Light Curve of WASP-18b (Concatenated)')
plt.show()

# Step 3: Use BLS to find periodic transits
# Define the time and flux for BLS
time -= np.min(time)  # Normalize time
flux_err = np.std(flux)  # Estimate flux error as standard deviation

# Initialize the BLS model
bls = at.BoxLeastSquares(time, flux)

# Define the period range for search
periods = np.linspace(0.8, 1.2, 10000)

# Adjust the duration for each period to avoid the error
# Transit duration is 5% of period, ensure it's always smaller than the period
durations = np.minimum(periods * 0.05, periods * 0.9)

bls_result = bls.power(periods, duration=durations)  # Use dynamic durations

# Step 4: Plot the periodogram
plt.figure(figsize=(10, 6))
plt.plot(bls_result.period, bls_result.power, color='b')
plt.xlabel('Period (days)')
plt.ylabel('Power (W)')
plt.title('BLS Periodogram')
plt.show()
```

```
# Step 5: Identify the strongest period
model = at.BoxLeastSquares(time, flux)
results = model.power(periods, durations)
index = np.argmax(results.power)

# Desired results:
# Period: 0.94 days
# Radius: 1.24 Jupiter radii
# Impact Parameter: ~0.32

best_period = bls_result.period[np.argmax(bls_result.power)]
print(f"Best-fit Period: {best_period:.4f} days")
t0 = results.transit_time[index]
duration = results.duration[index]

lc = lk.LightCurve(time=masked_time, flux=masked_flux)
folded_lc = lc.fold(period=best_period, epoch_time=t0)

# Shift the phase for three separate transits
phase = folded_lc.time.value
flux = folded_lc.flux.value

# Shift phases for three distinct transits
phase_shifted_pos1 = phase + best_period  # First shift
phase_shifted_pos2 = phase + 2 * best_period  # Second shift

# Concatenate the phases for three different transits
all_phases = np.concatenate((phase, phase_shifted_pos1, phase_shifted_pos2))
all_fluxes = np.concatenate((flux, flux, flux))

# Plot the three distinct transits
plt.scatter(all_phases, all_fluxes, s=5)
plt.xlabel('Phase (Radians)')
plt.ylabel('Normalized Flux (W/mˆ2')
plt.title('Three Separate Transits of WASP-18b')
plt.xlim(-1.5 * best_period, 3 * best_period)  # Adjust x-axis to fit three transits
plt.show()

# Scale the phase to stretch transits horizontally
```

```
scaled_phase = all_phases * 2.5 # Adjust the scale factor as needed

# Plot the horizontally stretched transits
plt.figure(figsize=(10, 6))
plt.scatter(scaled_phase, all_fluxes, s=10, alpha=0.5, label="Raw Data")

# Optional: Apply binning for clarity
bins = np.linspace(scaled_phase.min(), scaled_phase.max(), 500)  # Define bins
bin_indices = np.digitize(scaled_phase, bins)  # Assign phases to bins
binned_flux = [np.mean(all_fluxes[bin_indices == i]) for i in range(len(bins))]
plt.plot(bins, binned_flux, color='r', linewidth=1.5, label="Binned Data")

# Label and format the plot
plt.xlabel('Scaled Phase (Radians)')
plt.ylabel('Normalized Flux (W/mˆ2')
plt.title('(Horizontally Stretched) Transits of WASP-18b')
plt.xlim(-1, 5.8)  # Adjust x-axis range based on scaling
plt.legend()
plt.show()
```

LaTeX Typesetting

```
\documentclass[10pt,twocolumn]{article}
\usepackage[margin=0.7in]{geometry}
\usepackage{graphicx}
\usepackage{amsmath}
\usepackage{hyperref}
\usepackage{enumitem}
\usepackage{titlesec}  % For customizing section titles

\setlength{\columnsep}{0.25in}

% Customize \section* titles (Roman numerals) to be smaller and with reduced spacing before and after them
\titleformat{\section}[block]{\normalfont\bfseries\small}{\thesection}{1em}{}  % Smaller font, bold Roman numerals
\titlespacing*{\section}{0pt}{1ex}{1ex}  % Less vertical space before and after section title (adjusted to 1ex)

% Customize subsection titles to be smaller with controlled spacing
\titleformat{\subsection}[runin]{\normalfont\bfseries\small}{\thesubsection}{1em}{}
```

```
\titlespacing*{\subsection}{0pt}{0.5ex}{0.5ex}  % Less vertical space before and after subsection title

\title{\textbf{Class Project: Exploring WASP-18 b via the Transit Method}}
\author{Isaac Gutierrez - Class of 2024 \\ Ampersand: Gateway Expeditions Into Exoplanets - Fall 2024}
\date{}

\begin{document}

\maketitle

\section*{I. Transit Method Background}
Commencing as an observational method, the first scientific study of transits in the solar system was that of Mercury transiting the sun, observed by Pierre Gassendi in 1631 [1]. Since the discovery of HD 209458 b in 1999 as the first exoplanet to be discovered via the transit method, the method has soared to large popularity within the astrophysical community, leading to more than four thousand exoplanet discoveries via the transit method as of 2024. The idea of detecting planets via their shadows dates back centuries, but it became scientifically feasible in the late 20th century with advancements in photometry. Over time, space-based telescopes like Kepler and later TESS revolutionized the method, providing continuous and highly precise observations free from Earth's atmospheric interference, with the JWST mission providing a bright future for the detection of extrasolar planets via their transits.

\section*{II. Methodology of the Exoplanet Study}
\subsection*{A. Data Acquisition}
Reference [2], known as the Mikulski Archive for Space Telescopes, sources all relevant data in this project. 1), referring to the code assigned to step 1 of the class project (which will be a notation used for the code file signed to each respective part of the project), downloads all 2-minute cadence data on WASP-18 b from TESS, including all sectors of the relevant observation periods. The download comes in the form of multiple \texttt{.fits} files, which will allow the use of \texttt{astropy} features that will simplify 2) and onward.

\subsection*{B. Data Preprocessing:}
 2) processes the relevant data fields, masking unuseful data,
 concatenating the desired data, and plotting such data into a light curve, as shown in Figure 1.

\begin{figure}[!h]
    \centering
    \includegraphics[width=\columnwidth]{Figure 1.png}
```

```
   \caption{}
   \label{fig:lightcurve}
\end{figure}
```

Since data of poor quality is removed, an uncertainty is implemented in 2) on the concatenated data's flux only. Due to the time disparities in each observation period on WASP-18 b, the light curve appears compressed.

```
\subsection*{C. Rediscovering WASP-18 b:}
```
Through the box-least squares (BLS) \texttt{astropy} implementation, the following periodogram is generated in 3) (Figure 2).

```
\begin{figure}[!h]
   \centering
   \includegraphics[width=\columnwidth]{Figure 2.png}
   \caption{}
   \label{fig:placeholder2}
\end{figure}
```

Calculations of the periods of WASP-18 b's transits, along with their depths, are calculated via a "best-fit" period, or the most resonant peak from the generated periodogram. This best-fit period is calculated to be 0.94 days. Binning is used to remove noise and identify the flux of the transit period, with the transit depth calculated using the flux to be around 0.0067. With the given radius of WASP-18 as 1.319 solar radii, WASP-18 b's radius is calculated to be 0.11 solar radii, and the impact parameter of its transits is around 0.37 using the following equations, where $R_s$ is the radius of WASP-18, $R_p$ is the radius of WASP-18 b, $T$ is the transit depth, $a$ is the semi-major axis of WASP-18 b's orbit, $i$ is its inclination, and $b$ is the impact parameter:

$$
R_p = R_s \sqrt{T}, \quad b = \cos(i) \frac{a}{R_s}
$$

Uncertainty (flux error) is accounted for as the standard deviation of flux. This is also implemented in 4) to account for uncertainty.

```
\subsection*{D. The Primary Transits of WASP-18 b:}
```
The light curve of the best-fit period is folded in 4), which obtains the phase curve of the period. This is then plotted across three transits of WASP-18 b on its host star, as shown below, where the primary transit is highlighted.

\begin{figure}[!h]
  \centering
  \includegraphics[width=\columnwidth]{Figure 3.png}
  \caption{}
  \label{fig:placeholder3}
\end{figure}

Binning is applied to the graph, and the phase is scaled for clarity.

\subsection*{E. The Secondary Eclipse of WASP-18 b: }
The applied binning in 4) already shows distinct "secondary" transits in the light curve of WASP-18 b, but nonetheless 5) follows the desired instructions. Two of the three transits from 4) are highlighted, where the primary (most distinct) transits are removed from the light curve via masking. BLS is rerun, generating the following graph (Figure 4).

\begin{figure}[!h]
  \centering
  \includegraphics[width=\columnwidth]{Figure 4.png}
  \caption{}
  \label{fig:placeholder4}
\end{figure}

5) also bins the data to identify the secondary eclipse of WASP-18 b, which is most clear in the dip just before the leftmost "primary" transit in the light curve. By the scaled phase, this occurs between -1.0 radians and -0.5 radians, with a calculated depth of 0.0011. Through this given value, the dayside temperature of WASP-18 b can be calculated via the mathematical implementation in 5) to be around 3084 K.

\subsection*{F. Full Phase Curve of WASP-18 b:}
Via the phase curve generated by the periodogram data in 3), the full phase curve is plotted below with distinctions made to label the primary and secondary transits/eclipses. The phase is scaled to highlight the transits and focus on the landmarks of the phase curve. Primary transits are distinguished as purple regions, while secondary transits are distinguished as green regions, as shown below (Figure 5).

\begin{figure}[!h]
  \centering
  \includegraphics[width=\columnwidth]{Figure 5.png}
  \caption{}
  \label{fig:placeholder5}

\end{figure}

\section*{III. Results and Conclusion}
Via the data presented, the following quantities were inferred about WASP-18 b:
\begin{itemize}[noitemsep, topsep=0pt]
  \item \textbf{WASP-18 b's radius:} 0.11 solar radii.
  \item \textbf{WASP-18 b's dayside surface temperature:} ~3084 K.
\end{itemize}

These results are approximately (within a small margin of error) what is considered today as the accurate qualities of WASP-18 b in the astrophysical community. Through the photometric data from the Transiting Exoplanet Survey Satellite (TESS), the project utilized common software methods to make observations on a paradigm exoplanet, allowing us to make inferences on other details of the exoplanet through statistical analysis.

Furthermore, this methodology can be implemented to analyze other photometric data to make claims about other exoplanets, and serves as a strong example of how modern technologies continue to do so for the increasing discovery of new extrasolar planets in the cosmos.

\section*{References}
\begin{enumerate}[noitemsep, topsep=0pt]
  \item D. Briot and J. Schneider, ``Prehistory of Transit Searches,"
https://arxiv.org/abs/1803.06896.
  \item Space Telescope Science Institute, Barbara A. Mikulski Archive for Space Telescopes,
https://mast.stsci.edu/portal/Mashup/Clients/Mast/
  Portal.html
\end{enumerate}

\end{document}