turfpy

Omkar Mestry, Sachin Kharude

Jul 24, 2021

GETTING STARTED

A Python library for performing geospatial data analysis which reimplements turf.js.

CHAPTER

ONE

INSTALLATION

1.1 Using pip

```
pip install turfpy
```

1.2 Using conda

```
conda install -c conda-forge turfpy
```

CHAPTER

TWO

BEARING

Takes two Points and finds the geographic bearing between them.

2.1 Example

```
from turfpy import measurement
from geojson import Point, Feature

start = Feature(geometry=Point((-75.343, 39.984)))
end = Feature(geometry=Point((-75.534, 39.123)))
measurement.bearing(start, end)
```

```
-170.23304913492177
```

CHAPTER

THREE

DISTANCE

Calculates distance between two Points.

3.1 Example

```
from turfpy import measurement
from geojson import Point, Feature
start = Feature(geometry=Point((-75.343, 39.984)))
end = Feature(geometry=Point((-75.534, 39.123)))
measurement.distance(start,end)
```

```
97.12922118967835
```

CHAPTER

FOUR

AREA

Calculates the area of the input GeoJSON object.

4.1 Example

```
from turfpy.measurement import area
from geojson import Feature, FeatureCollection

geometry_1 = {
    "coordinates": [[[0, 0], [0, 10], [10, 10], [10, 0], [0, 0]]],
    "type": "Polygon",
}
geometry_2 = {
    "coordinates": [
        [[2.38, 57.322], [23.194, -20.28], [-120.43, 19.15], [2.38, 57.322]]
    ],
    "type": "Polygon",
}
feature_1 = Feature(geometry=geometry_1)
feature_2 = Feature(geometry=geometry_2)
feature_collection = FeatureCollection([feature_1, feature_2])

area(feature_collection)
```

```
56837434206665.02
```

4.2 Interactive Example

```
from turfpy.measurement import area
from geojson import Feature, FeatureCollection
from ipyleaflet import Map, GeoJSON, basemaps, basemap_to_tiles, WidgetControl
from ipywidgets import HTML

geometry_1 = {
    "coordinates": [[[0, 0], [0, 10], [10, 10], [10, 0], [0, 0]]],
    "type": "Polygon",
}
```

(continues on next page)

(continued from previous page)

```
geometry_2 = {
    "coordinates": [
        [[2.38, 57.322], [23.194, -20.28], [-120.43, 19.15], [2.38, 57.322]]
    ],
    "type": "Polygon",
}
feature_1 = Feature(geometry=geometry_1)
feature_2 = Feature(geometry=geometry_2)
feature_collection = FeatureCollection([feature_1, feature_2])
geo_json = GeoJSON(data=feature_collection)
watercolor = basemap_to_tiles(basemaps.Stamen.Watercolor)

m = Map(layers=(watercolor,), center=[20.04303061200023, -11.832275390625002], zoom=2)

m.add_layer(geo_json)

html = HTML()
html.layout.margin = "0px 20px 10px 20px"
html.value = """
        <h4>Area in meter sqaure: {}</h4>
        <h4>measurement.area(feature_collection)</h4>
    """.format(
    area(feature_collection)
)
control = WidgetControl(widget=html, position="topright")
m.add_control(control)
m
```

CHAPTER

FIVE

BBOX

Generate bounding box coordinates for given geojson.

5.1 Example

```
from turfpy.measurement import bbox
from geojson import Polygon

p = Polygon([[(2.38, 57.322), (23.194, -20.28), (-120.43, 19.15), (2.38, 57.322)]])
bbox(p)
```

```
[-120.43, -20.28, 23.194, 57.322]
```

5.2 Interactive Example

```
from turfpy.measurement import bbox
from geojson import Polygon
from geojson import Feature
from ipyleaflet import Map, GeoJSON, WidgetControl
from ipywidgets import HTML

p = Polygon([[(2.38, 57.322), (23.194, -20.28), (-120.43, 19.15), (2.38, 57.322)]])


geo_json = GeoJSON(data=Feature(geometry=p))

m = Map(center=[20.04303061200023, -11.832275390625002], zoom=2)

m.add_layer(geo_json)

html = HTML()
html.layout.margin = "0px 20px 10px 20px"
html.value = """
        <h4>Bounding Box for given geojson</h4>
        <h4>{}</h4>
    """.format(
    bbox(p)
```

(continues on next page)

(continued from previous page)

```
)
control = WidgetControl(widget=html, position="topright")
m.add_control(control)

m
```

CHAPTER

SIX

BBOX POLYGON

Generate a Polygon Feature for the bounding box generated using bbox.

6.1 Example

```
import json

from turfpy.measurement import bbox_polygon, bbox
from geojson import Polygon

p = Polygon([[(2.38, 57.322), (23.194, -20.28), (-120.43, 19.15), (2.38, 57.322)]])
bb = bbox(p)
print(json.dumps(bbox_polygon(bb), indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      [
        [
          -120.43,
          -20.28
        ],
        [
          23.194,
          -20.28
        ],
        [
          23.194,
          57.322
        ],
        [
          -120.43,
          57.322
        ],
        [
          -120.43,
          -20.28
        ]
      ]
```

(continues on next page)

(continued from previous page)

```
    ],
    "type": "Polygon"
  },
  "properties": {},
  "type": "Feature"
}
```

6.2 Interactive Example

```
from turfpy.measurement import bbox_polygon, bbox
from geojson import Polygon, Feature
from ipyleaflet import Map, GeoJSON, WidgetControl, LayersControl

p = Polygon([[(2.38, 57.322), (23.194, -20.28), (-120.43, 19.15), (2.38, 57.322)]])
bb = bbox(p)
geo_json = GeoJSON(name="Geojson", data=Feature(geometry=p))
bbox_polygon_geojson = GeoJSON(
    name="Bounding Box Polygon", data=bbox_polygon(bb), style={"color": "red"}
)

m = Map(center=[20.04303061200023, -11.832275390625002], zoom=2)

control = LayersControl(position="topright")
m.add_control(control)

m.add_layer(geo_json)
m.add_layer(bbox_polygon_geojson)
m
```

CHAPTER

SEVEN

CENTER

Takes a Feature or FeatureCollection and returns the absolute center point of all features.

7.1 Example

```
import json
from turfpy.measurement import center
from geojson import Feature, FeatureCollection, Point

f1 = Feature(geometry=Point((-97.522259, 35.4691)))
f2 = Feature(geometry=Point((-97.502754, 35.463455)))
f3 = Feature(geometry=Point((-97.508269, 35.463245)))
feature_collection = FeatureCollection([f1, f2, f3])
print(json.dumps(center(feature_collection), indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      -97.512507,
      35.466172
    ],
    "type": "Point"
  },
  "properties": {},
  "type": "Feature"
}
```

7.2 Interactive Example

```
from turfpy.measurement import center
from geojson import Feature, FeatureCollection, Point
from ipyleaflet import Map, GeoJSON, WidgetControl, LayersControl, CircleMarker

f1 = Feature(geometry=Point((-97.522259, 35.4691)))
f2 = Feature(geometry=Point((-97.502754, 35.463455)))
f3 = Feature(geometry=Point((-97.508269, 35.463245)))
feature_collection = FeatureCollection([f1, f2, f3])
```

(continues on next page)

(continued from previous page)

```
m = Map(center=[35.467146770097315, -97.50865470618012], zoom=15)

geo_json = GeoJSON(name="Geojson", data=feature_collection)

centre_coord = center(feature_collection)["geometry"]["coordinates"]
circle_marker = CircleMarker(name="Center")
circle_marker.location = (centre_coord[1], centre_coord[0])
circle_marker.radius = 30
circle_marker.color = "red"

control = LayersControl(position="topright")
m.add_control(control)

m.add_layer(geo_json)
m.add_layer(circle_marker)
m
```

CHAPTER

EIGHT

ENVELOP

Takes any number of features and returns a rectangular Polygon that encompasses all vertices.

8.1 Example

```
import json
from turfpy.measurement import envelope
from geojson import Feature, FeatureCollection, Point

f1 = Feature(geometry=Point((-97.522259, 35.4691)))
f2 = Feature(geometry=Point((-97.502754, 35.463455)))
f3 = Feature(geometry=Point((-97.508269, 35.463245)))
feature_collection = FeatureCollection([f1, f2, f3])
print(json.dumps(envelope(feature_collection), indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      [
        [
          -97.522259,
          35.463245
        ],
        [
          -97.502754,
          35.463245
        ],
        [
          -97.502754,
          35.4691
        ],
        [
          -97.522259,
          35.4691
        ],
        [
          -97.522259,
          35.463245
        ]
```

(continues on next page)

(continued from previous page)

```
      ]
    ],
    "type": "Polygon"
  },
  "properties": {},
  "type": "Feature"
}
```

8.2 Interactive Example

```
from turfpy.measurement import envelope
from geojson import Feature, FeatureCollection, Point
from ipyleaflet import Map, GeoJSON, WidgetControl, LayersControl

f1 = Feature(geometry=Point((-97.522259, 35.4691)))
f2 = Feature(geometry=Point((-97.502754, 35.463455)))
f3 = Feature(geometry=Point((-97.508269, 35.463245)))
feature_collection = FeatureCollection([f1, f2, f3])


m = Map(center=[35.467146770097315, -97.50865470618012], zoom=15)

geo_json = GeoJSON(name="Geojson", data=feature_collection)
envelope_geojson = GeoJSON(
    name="Envelope", data=envelope(feature_collection), style={"color": "red"}
)

control = LayersControl(position="topright")
m.add_control(control)

m.add_layer(geo_json)
m.add_layer(envelope_geojson)
m
```

CHAPTER

NINE

AREA

Takes a geojson and measures its length in the specified units.

9.1 Example

```
from turfpy.measurement import length
from geojson import LineString
ls = LineString([(115, -32), (131, -22), (143, -25), (150, -34)])
length(ls)
```

```
4407.93912491419
```

9.2 Interactive Example

```
from turfpy.measurement import length
from geojson import LineString, Feature
from ipyleaflet import Map, GeoJSON, WidgetControl
from ipywidgets import HTML

ls = LineString([(115, -32), (131, -22), (143, -25), (150, -34)])

m = Map(center=[-25.52664223616833, 143.44917297363284], zoom=4)

geo_json = GeoJSON(name="Geojson", data=Feature(geometry=ls))

m.add_layer(geo_json)

html = HTML()
html.layout.margin = "0px 20px 10px 20px"
html.value = """
        <h4>Length of the given geojson in meters</h4>
        <h4>{}</h4>
    """.format(
    length(ls, units="m")
)

control = WidgetControl(widget=html, position="topright")
```

(continues on next page)

(continued from previous page)

```
m.add_control(control)

m
```

CHAPTER

TEN

DESTINATION

Takes a Point and calculates the location of a destination point given a distance in degrees, radians, miles, or kilometers and bearing in degrees.

10.1 Example

```
import json
from turfpy.measurement import destination
from geojson import Point, Feature

origin = Feature(geometry=Point([-75.343, 39.984]))
distance = 50
bearing = 90
options = {'units': 'mi'}
print(json.dumps(destination(origin,distance,bearing,options), indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      -74.398588,
      39.980168
    ],
    "type": "Point"
  },
  "properties": {},
  "type": "Feature"
}
```

10.2 Interactive Example

```
from turfpy.measurement import destination
from geojson import Point, Feature
from ipyleaflet import Map, GeoJSON, WidgetControl, LayersControl, CircleMarker
from ipywidgets import HTML

origin = Feature(geometry=Point([-75.343, 39.984]))
distance = 50
```

(continues on next page)

(continued from previous page)

```
bearing = 90
options = {"units": "mi"}

m = Map(center=[39.98304755619415, -74.67888951301576], zoom=9)


geo_json = GeoJSON(name="Start Point", data=origin)

centre_coord = destination(origin, distance, bearing, options)["geometry"][
    "coordinates"
]
circle_marker = CircleMarker(name="End Point")
circle_marker.location = (centre_coord[1], centre_coord[0])
circle_marker.radius = 10
circle_marker.color = "red"


m.add_layer(geo_json)
m.add_layer(circle_marker)

html = HTML()
html.layout.margin = "0px 20px 10px 20px"
html.value = """
        <h4>Point which is at 90 degrees bearing and 50 miles in that direction</h4>
    """

control = WidgetControl(widget=html, position="topright")
m.add_control(control)

control = LayersControl(position="topright")
m.add_control(control)

m
```

CHAPTER

ELEVEN

CENTROID

Takes one or more features and calculates the centroid using the mean of all vertices.

11.1 Example

```
import json
from turfpy.measurement import centroid
from geojson import Polygon
polygon = Polygon([[(-81, 41), (-88, 36), (-84, 31), (-80, 33), (-77, 39), (-81, 41)]])
print(json.dumps(centroid(polygon), indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      -81.833333,
      36.833333
    ],
    "type": "Point"
  },
  "properties": {},
  "type": "Feature"
}
```

11.2 Interactive Example

```
from turfpy.measurement import centroid
from geojson import Polygon, Feature
from ipyleaflet import Map, GeoJSON, WidgetControl, LayersControl

polygon = Polygon([[(-81, 41), (-88, 36), (-84, 31), (-80, 33), (-77, 39), (-81, 41)]])

m = Map(center=[36.198988385375806, -79.6392059326172], zoom=5)

geo_json = GeoJSON(name="Geojson", data=Feature(geometry=polygon))
centroid_geojson = GeoJSON(name="Centroid", data=centroid(polygon))

control = LayersControl(position="topright")
```

(continues on next page)

(continued from previous page)

```
m.add_control(control)

m.add_layer(geo_json)
m.add_layer(centroid_geojson)
m
```

CHAPTER

TWELVE

ALONG

This function is used identify a Point at a specified distance along a LineString.

12.1 Example

```
import json
from turfpy.measurement import along
from geojson import LineString
ls = LineString([(-83, 30), (-84, 36), (-78, 41)])
print(json.dumps(along(ls,200,'mi'), indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      -83.460865,
      32.86781
    ],
    "type": "Point"
  },
  "properties": {},
  "type": "Feature"
}
```

12.2 Interactive Example

```
from turfpy.measurement import along, length
from geojson import LineString, Feature
from ipyleaflet import Map, GeoJSON, WidgetControl, Marker
from ipywidgets import FloatSlider

ls = LineString([(-83, 30), (-84, 36), (-78, 41)])


m = Map(center=[35.47241402319959, -80.11693954467775], zoom=5)
marker = Marker(location=[30, -83])
m.add_layer(marker)
```

(continues on next page)

(continued from previous page)

```
def on_value_change(change):
    global marker
    new_point = along(ls, change["new"], "mi")
    marker.location = new_point["geometry"]["coordinates"][::-1]


style = {"description_width": "initial"}
slider = FloatSlider(
    description="Marker position:",
    min=0,
    max=length(ls, units="mi"),
    value=0,
    style=style,
)
slider.observe(on_value_change, names="value")

widget_control1 = WidgetControl(widget=slider, position="topright")
m.add_control(widget_control1)


geo_json = GeoJSON(name="Geojson", data=Feature(geometry=ls))

m.add_layer(geo_json)

m
```

CHAPTER

THIRTEEN

MIDPOINT

Get midpoint between any the two points.

13.1 Example

```
import json
from turfpy.measurement import midpoint
from geojson import Point, Feature
point1 = Feature(geometry=Point([144.834823, -37.771257]))
point2 = Feature(geometry=Point([145.14244, -37.830937]))
print(json.dumps(midpoint(point1, point2), indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      144.988569,
      -37.801197
    ],
    "type": "Point"
  },
  "properties": {},
  "type": "Feature"
}
```

13.2 Interactive Example

```
from turfpy.measurement import midpoint
from geojson import Point, Feature
from ipyleaflet import Map, GeoJSON, LayersControl, CircleMarker


point1 = Feature(geometry=Point([144.834823, -37.771257]))
point2 = Feature(geometry=Point([145.14244, -37.830937]))

m = Map(center=[-37.80415546165204, 145.0286749005318], zoom=11)

start_geo_json = GeoJSON(name="Start Point", data=point1)
```

(continues on next page)

(continued from previous page)

```
end_geo_json = GeoJSON(name="End Point", data=point2)
m.add_layer(start_geo_json)
m.add_layer(end_geo_json)

midpoint_coord = midpoint(point1, point2)["geometry"]["coordinates"]
circle_marker = CircleMarker(name="Midpoint")
circle_marker.location = (midpoint_coord[1], midpoint_coord[0])
circle_marker.radius = 10
circle_marker.color = "red"
m.add_layer(circle_marker)

control = LayersControl(position="topright")
m.add_control(control)

m
```

CHAPTER

FOURTEEN

NEAREST POINT

Takes a reference Point Feature and FeatureCollection of point features and returns the point from the FeatureCollection closest to the reference Point Feature.

14.1 Example

```
import json
from turfpy.measurement import nearest_point
from geojson import Point, Feature, FeatureCollection
f1 = Feature(geometry=Point([28.96991729736328,41.01190001748873]))
f2 = Feature(geometry=Point([28.948459, 41.024204]))
f3 = Feature(geometry=Point([28.938674, 41.013324]))
fc = FeatureCollection([f1, f2 ,f3])
t = Feature(geometry=Point([28.973865, 41.011122]))
print(json.dumps(nearest_point(t ,fc), indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      28.969917,
      41.0119
    ],
    "type": "Point"
  },
  "properties": {
    "distanceToPoint": 0.34236818771791105,
    "featureIndex": 0
  },
  "type": "Feature"
}
```

14.2 Interactive Example

```
from turfpy.measurement import nearest_point
from geojson import Point, Feature, FeatureCollection
from ipyleaflet import Map, GeoJSON, LayersControl

f1 = Feature(geometry=Point([28.96991729736328, 41.01190001748873]))
f2 = Feature(geometry=Point([28.948459, 41.024204]))
f3 = Feature(geometry=Point([28.938674, 41.013324]))
fc = FeatureCollection([f1, f2, f3])
t = Feature(geometry=Point([28.973865, 41.011122]))


m = Map(center=[41.01656246584522, 28.959988430142406], zoom=14)

geo_json = GeoJSON(name="Feature Collection", data=FeatureCollection([f2, f3]))

ref_geo_json = GeoJSON(name="Reference Point", data=t)

m.add_layer(geo_json)

m.add_layer(ref_geo_json)

near_geojson = GeoJSON(name="Nearest Point", data=nearest_point(t, fc))
m.add_layer(near_geojson)

control = LayersControl(position="topright")
m.add_control(control)

m
```

CHAPTER

FIFTEEN

POINT ON FEATURE

Takes a Feature or FeatureCollection and returns a Point guaranteed to be on the surface of the feature.

15.1 Example

```
import json
from turfpy.measurement import point_on_feature
from geojson import  Polygon, Feature
point = Polygon([[(116, -36), (131, -32), (146, -43), (155, -25), (133, -9), (111, -22),␣
↪(116, -36)]])
feature = Feature(geometry=point)
print(json.dumps(point_on_feature(feature), indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      129.714286,
      -29.0
    ],
    "type": "Point"
  },
  "properties": {},
  "type": "Feature"
}
```

15.2 Interactive Example

```
from turfpy.measurement import point_on_feature
from geojson import  Polygon, Feature
from ipyleaflet import Map, GeoJSON, LayersControl

point = Polygon([[(116, -36), (131, -32), (146, -43), (155, -25), (133, -9), (111, -22),␣
↪(116, -36)]])
feature = Feature(geometry=point)


m = Map(center=[-25.743003105825967, 135.92525482177737], zoom=4)
```

(continues on next page)

(continued from previous page)

```
geo_json = GeoJSON(name='Feature', data=feature)

m.add_layer(geo_json)

point_geojson = GeoJSON(name='Point on Feature', data=point_on_feature(feature))

m.add_layer(point_geojson)

control = LayersControl(position='topright')
m.add_control(control)

m
```

CHAPTER

SIXTEEN

POINT IN POLYGON

Takes a Point or a Point Feature and Polygon or Polygon Feature as input and returns True if Point is in given Feature.

16.1 Example

```
from turfpy.measurement import boolean_point_in_polygon
from geojson import Point, MultiPolygon, Feature
point = Feature(geometry=Point([-77, 44]))
polygon = Feature(geometry=MultiPolygon([([(-81, 41), (-81, 47), (-72, 47), (-72, 41), (-
↪81, 41)],),
([(3.78, 9.28), (-130.91, 1.52), (35.12, 72.234), (3.78, 9.28)],)]))
boolean_point_in_polygon(point, polygon)
```

```
True
```

16.2 Interactive Example

```
from turfpy.measurement import boolean_point_in_polygon
from geojson import Point, MultiPolygon, Feature
from ipyleaflet import Map, GeoJSON, LayersControl

point = Feature(geometry=Point([-77, 44]))
polygon = Feature(
    geometry=MultiPolygon(
        [
            ([(-81, 41), (-81, 47), (-72, 47), (-72, 41), (-81, 41)],),
            ([(3.78, 9.28), (-130.91, 1.52), (35.12, 72.234), (3.78, 9.28)],),
        ]
    )
)
boolean_point_in_polygon(point, polygon)

m = Map(center=[46.57868671298067, -40.91583251953126], zoom=2)

geo_json = GeoJSON(name="MultiPolygon Feature", data=polygon)

m.add_layer(geo_json)
```

(continues on next page)

(continued from previous page)

```
point_geojson = GeoJSON(name="Point in Polygon", data=point)

m.add_layer(point_geojson)

control = LayersControl(position="topright")
m.add_control(control)

m
```

CHAPTER

SEVENTEEN

POINT TO LINE DISTANCE

Returns the minimum distance between a Point and any segment of the LineString.

17.1 Example

```
from turfpy.measurement import point_to_line_distance
from geojson import LineString, Point, Feature
point = Feature(geometry=Point([0, 0]))
linestring = Feature(geometry=LineString([(1, 1),(-1, 1)]))
point_to_line_distance(point, linestring)
```

```
111.19508023353292
```

17.2 Interactive Example

```
from turfpy.measurement import point_to_line_distance
from geojson import LineString, Point, Feature
from ipyleaflet import Map, GeoJSON, LayersControl, WidgetControl
from ipywidgets import HTML

point = Feature(geometry=Point([0, 0]))
linestring = Feature(geometry=LineString([(1, 1),(-1, 1)]))

m = Map(center=[0.5427636983179688, 0.3891992568969727], zoom=8)

geo_json = GeoJSON(name='Point', data=point)

m.add_layer(geo_json)

point_geojson = GeoJSON(name='Linestring', data=linestring)

m.add_layer(point_geojson)

html = HTML()
html.layout.margin = '0px 20px 10px 20px'
html.value = '''
        <h4>Minimum distance between the Point and the LineString in meters</h4>
```

(continues on next page)

(continued from previous page)

```
        <h4>{}</h4>
    '''.format(point_to_line_distance(point, linestring, units='m'))

control = WidgetControl(widget=html, position='topright')
m.add_control(control)

m
```

CHAPTER

EIGHTEEN

RHUMB BEARING

Takes two points and finds the bearing angle between them along a Rhumb line i.e. the angle measured in degrees start the north line (0 degrees).

18.1 Example

```
from turfpy.measurement import rhumb_bearing
from geojson import Feature, Point
start = Feature(geometry=Point([-75.343, 39.984]))
end = Feature(geometry=Point([-75.534, 39.123]))
rhumb_bearing(start, end, True)
```

```
9.705824644274514
```

CHAPTER

NINETEEN

RHUMB DESTINATION

Returns the destination Point having travelled the given distance along a Rhumb line from the origin Point with the (varant) given bearing.

19.1 Example

```
import json
from turfpy.measurement import rhumb_destination
from geojson import Point, Feature
start = Feature(geometry=Point([-75.343, 39.984]), properties={"marker-color": "F00"})
distance = 50
bearing = 90
rd = rhumb_destination(start, distance, bearing, {'units':'mi', 'properties': {"marker-
↪color": "F00"}})
print(json.dumps(rd, indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      -74.398553,
      39.984
    ],
    "type": "Point"
  },
  "properties": {
    "marker-color": "F00"
  },
  "type": "Feature"
}
```

CHAPTER

TWENTY

RHUMB DISTANCE

Calculates the distance along a rhumb line between two points in degrees, radians, miles, or kilometers.

20.1 Example

```
from turfpy.measurement import rhumb_distance
from geojson import Point, Feature
start = Feature(geometry=Point([-75.343, 39.984]))
end = Feature(geometry=Point([-75.534, 39.123]))
rhumb_distance(start, end,'mi')
```

```
60.353311281256794
```

CHAPTER

TWENTYONE

SQUARE

Takes a bounding box and calculates the minimum square bounding box that would contain the input.

21.1 Example

```
from turfpy.measurement import square
bbox = [-20, -20, -15, 0]
square(bbox)
```

```
[-27.5, -20, -7.5, 0]
```

21.2 Interactive Example

```
from turfpy.measurement import square, bbox_polygon
from geojson import Point, Feature
from ipyleaflet import Map, GeoJSON, LayersControl

bbox = [-20, -20, -15, 0]

sqaure_geo_json = GeoJSON(
    name="Sqaure for the given Bounding Box",
    data=bbox_polygon(square(bbox)),
    style={"color": "red"},
)
bbox_polygon_geojson = GeoJSON(name="Bounding Box", data=bbox_polygon(bbox))

m = Map(center=[-8.484257262005082, -11.58611297607422], zoom=4)

control = LayersControl(position="topright")
m.add_control(control)

m.add_layer(sqaure_geo_json)
m.add_layer(bbox_polygon_geojson)
m
```

CHAPTER

TWENTYTWO

POINTS WITHIN POLYGON

Takes two inputs Point/Points and Polygon(s)/MultiPolygon(s) and returns all the Points with in Polygon(s)/MultiPolygon(s).

22.1 Interactive Example

```
from geojson import Feature, FeatureCollection, Point, Polygon
from turfpy.measurement import points_within_polygon
from ipyleaflet import Map, GeoJSON

p1 = Feature(geometry=Point((-46.6318, -23.5523)))
p2 = Feature(geometry=Point((-46.6246, -23.5325)))
p3 = Feature(geometry=Point((-46.6062, -23.5513)))
p4 = Feature(geometry=Point((-46.663, -23.554)))
p5 = Feature(geometry=Point((-46.643, -23.557)))

points = FeatureCollection([p1, p2, p3, p4, p5])

poly = Polygon(
    [
        [
            (-46.653, -23.543),
            (-46.634, -23.5346),
            (-46.613, -23.543),
            (-46.614, -23.559),
            (-46.631, -23.567),
            (-46.653, -23.560),
            (-46.653, -23.543),
        ]
    ]
)

m = Map(center=(-23.5523, -46.6318), zoom=13)
fc = FeatureCollection([p1, p2, p3, p4, p5, poly])

geo_json = GeoJSON(
    data=fc,
    style={"opacity": 1, "dashArray": "9", "fillOpacity": 0.3, "weight": 1},
    hover_style={"color": "green", "dashArray": "0", "fillOpacity": 0.5},
```

(continues on next page)

(continued from previous page)

```
)
m.add_layer(geo_json)

result = points_within_polygon(points, poly)

data = result.copy()
data["features"].append(Feature(geometry=poly))

m = Map(center=(-23.5523, -46.6318), zoom=13)

geo_json2 = GeoJSON(
    data=data,
    style={"opacity": 1, "dashArray": "9", "fillOpacity": 0.3, "weight": 1},
    hover_style={"color": "green", "dashArray": "0", "fillOpacity": 0.5},
)
m.add_layer(geo_json2)
m
```

CHAPTER

TWENTYTHREE

CIRCLE

Takes a Point and calculates the circle polygon given a radius in degrees, radians, miles, or kilometers; and steps for precision.

23.1 Example

```
import json
from geojson import Feature, Point
from turfpy.transformation import circle

center = Feature(geometry=Point((19.0760, 72.8777)))
cc = circle(center, radius=5, steps=10)
print(json.dumps(cc, indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      [
        [
          19.076,
          72.922666
        ],
        [
          18.986041,
          72.914058
        ],
        [
          18.93063,
          72.891543
        ],
        [
          18.930858,
          72.863753
        ],
        [
          18.986411,
          72.841302
        ],
        [
```

(continues on next page)

(continued from previous page)


```
          19.076,
          72.832734
        ],
        [
          19.165589,
          72.841302
        ],
        [
          19.221142,
          72.863753
        ],
        [
          19.22137,
          72.891543
        ],
        [
          19.165959,
          72.914058
        ],
        [
          19.076,
          72.922666
        ]
      ]
    ],
    "type": "Polygon"
  },
  "properties": {},
  "type": "Feature"
}
```


23.2 Interactive Example

```
from geojson import Point, Feature
from ipyleaflet import Map, GeoJSON
from turfpy.transformation import circle

center = Feature(geometry=Point((-75.343, 39.984)))

geo_json = GeoJSON(data=circle(center, radius=5, steps=10, units='km'))

m = Map(center=[39.978756161038504, -75.32421022653581], zoom=11)

m.add_layer(geo_json)

m
```

CHAPTER

TWENTYFOUR

BBOX CLIP

Takes a Feature or geometry and a bbox and clips the feature to the bbox.

24.1 Example

```
import json
from turfpy.transformation import bbox_clip
from geojson import Feature
f = Feature(geometry={"coordinates": [[[2, 2], [8, 4],
[12, 8], [3, 7], [2, 2]]], "type": "Polygon"})
bbox = [0, 0, 10, 10]
bc = bbox_clip(f, bbox)
print(json.dumps(bc, indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      [
        [
          10.0,
          7.777778
        ],
        [
          10.0,
          6.0
        ],
        [
          8.0,
          4.0
        ],
        [
          2.0,
          2.0
        ],
        [
          3.0,
          7.0
        ],
        [
```

(continues on next page)

(continued from previous page)

```
          10.0,
          7.777778
        ]
      ]
    ],
    "type": "Polygon"
  },
  "properties": {},
  "type": "Feature"
}
```

24.2 Interactive Example

```
from geojson import Feature
from ipyleaflet import Map, GeoJSON, LayersControl
from turfpy.transformation import bbox_clip, bbox_polygon


f = Feature(
    geometry={
        "coordinates": [[[2, 2], [8, 4], [12, 8], [3, 7], [2, 2]]],
        "type": "Polygon",
    }
)

bbox = [0, 0, 10, 10]

geo_json = GeoJSON(name="Polygon", data=f)

bbox_polygon_geojson = GeoJSON(
    name="Bounding Box Polygon", data=bbox_polygon(bbox), style={"color": "green"}
)

cliped_geojson = GeoJSON(
    name="Clipped Polygon", data=bbox_clip(f, bbox), style={"color": "red"}
)


m = Map(center=[4.889835742990713, 5.82601547241211], zoom=5)

m.add_layer(geo_json)
m.add_layer(bbox_polygon_geojson)
m.add_layer(cliped_geojson)

control = LayersControl(position="topright")
m.add_control(control)

m
```

CHAPTER

TWENTYFIVE

BEZIER SPLINE

Takes a line and returns a curved version by applying a Bezier spline algorithm.

25.1 Example

```
import json
from geojson import LineString, Feature
from turfpy.transformation import bezier_spline
ls = LineString([(-76.091308, 18.427501),
                     (-76.695556, 18.729501),
                     (-76.552734, 19.40443),
                     (-74.61914, 19.134789),
                     (-73.652343, 20.07657),
                     (-73.157958, 20.210656)])
f = Feature(geometry=ls)
bs = bezier_spline(f)
print(json.dumps(bs, indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      [
        -76.091308,
        18.427501
      ],
      [
        -76.091346,
        18.427508
      ],
      [
        -76.091459,
        18.427529
      ],
      [
        -76.091647,
        18.427565
      ],
      [
        -76.091908,
```

(continues on next page)

(continued from previous page)

```
        18.427614
      ],
      [
        -76.092243,
        18.427678
      ],
      [
        -76.09265,
        18.427756
      ],
      [
        -76.093129,
        18.427849
      ],
      [
        -76.093679,
        18.427955
      ],
      [
        -76.0943,
        18.428076
      ],
      [
        -76.10558,
        18.430352
      ],
      [
        -76.106992,
        18.430645
      ],
      [
        -76.108466,
        18.430953
      ],
      [
        -76.110001,
        18.431275
      ],
      [
        -76.111596,
        18.431612
      ],
      [
        -76.11325,
        18.431963
      ],
      [
        -76.114963,
        18.432328
      ],
      [
        -76.116735,
```

(continues on next page)

(continued from previous page)

```
        18.432709
      ],
      [
        -76.118563,
        18.433103
      ],
      [
        -76.120449,
        18.433513
      ],
      [
        -76.144737,
        18.43898
      ],
      [
        -76.147249,
        18.439565
      ],
      [
        -76.14981,
        18.440164
      ],
      [
        -76.152418,
        18.440779
      ],
      [
        -76.155072,
        18.441408
      ],
      [
        -76.157771,
        18.442052
      ],
      [
        -76.160516,
        18.442711
      ],
      [
        -76.163306,
        18.443385
      ],
      [
        -76.166139,
        18.444074
      ],
      [
        -76.169015,
        18.444777
      ],
      [
        -76.203294,
```

(continues on next page)

(continued from previous page)

```
        18.453497
      ],
      [
        -76.206632,
        18.45438
      ],
      [
        -76.210005,
        18.455277
      ],
      [
        -76.213411,
        18.45619
      ],
      [
        -76.21685,
        18.457117
      ],
      [
        -76.220321,
        18.45806
      ],
      [
        -76.223823,
        18.459018
      ],
      [
        -76.227356,
        18.45999
      ],
      [
        -76.230919,
        18.460978
      ],
      [
        -76.234512,
        18.461981
      ],
      [
        -76.275764,
        18.474017
      ],
      [
        -76.279654,
        18.475203
      ],
      [
        -76.283565,
        18.476403
      ],
      [
        -76.287496,
```

(continues on next page)

(continued from previous page)

```
        18.47762
      ],
      [
        -76.291445,
        18.478851
      ],
      [
        -76.295413,
        18.480098
      ],
      [
        -76.299398,
        18.48136
      ],
      [
        -76.303401,
        18.482638
      ],
      [
        -76.307419,
        18.483931
      ],
      [
        -76.311453,
        18.485239
      ],
      [
        -76.356662,
        18.500652
      ],
      [
        -76.36083,
        18.502146
      ],
      [
        -76.365004,
        18.503656
      ],
      [
        -76.369185,
        18.505182
      ],
      [
        -76.37337,
        18.506723
      ],
      [
        -76.377561,
        18.508279
      ],
      [
        -76.381755,
```

(continues on next page)

(continued from previous page)

```
        18.509852
      ],
      [
        -76.385952,
        18.51144
      ],
      [
        -76.390152,
        18.513043
      ],
      [
        -76.394354,
        18.514663
      ],
      [
        -76.440502,
        18.533515
      ],
      [
        -76.444673,
        18.535324
      ],
      [
        -76.448836,
        18.537148
      ],
      [
        -76.452992,
        18.538988
      ],
      [
        -76.45714,
        18.540845
      ],
      [
        -76.461278,
        18.542717
      ],
      [
        -76.465407,
        18.544605
      ],
      [
        -76.469525,
        18.546509
      ],
      [
        -76.473632,
        18.548429
      ],
      [
        -76.477728,
```

(continues on next page)

(continued from previous page)

```
        18.550365
      ],
      [
        -76.521797,
        18.572719
      ],
      [
        -76.525697,
        18.574847
      ],
      [
        -76.529575,
        18.576992
      ],
      [
        -76.533433,
        18.579153
      ],
      [
        -76.537268,
        18.58133
      ],
      [
        -76.54108,
        18.583523
      ],
      [
        -76.544869,
        18.585733
      ],
      [
        -76.548634,
        18.587959
      ],
      [
        -76.552374,
        18.590201
      ],
      [
        -76.556088,
        18.592459
      ],
      [
        -76.595062,
        18.618376
      ],
      [
        -76.598416,
        18.62083
      ],
      [
        -76.601735,
```

(continues on next page)

(continued from previous page)

```
        18.623301
      ],
      [
        -76.60502,
        18.625788
      ],
      [
        -76.608268,
        18.628291
      ],
      [
        -76.61148,
        18.630811
      ],
      [
        -76.614655,
        18.633348
      ],
      [
        -76.617792,
        18.635901
      ],
      [
        -76.62089,
        18.638471
      ],
      [
        -76.623949,
        18.641057
      ],
      [
        -76.65481,
        18.670599
      ],
      [
        -76.657344,
        18.673384
      ],
      [
        -76.65983,
        18.676187
      ],
      [
        -76.662268,
        18.679006
      ],
      [
        -76.664655,
        18.681841
      ],
      [
        -76.666992,
```

(continues on next page)

(continued from previous page)

```
        18.684694
      ],
      [
        -76.669279,
        18.687563
      ],
      [
        -76.671514,
        18.690449
      ],
      [
        -76.673696,
        18.693352
      ],
      [
        -76.675825,
        18.696272
      ],
      [
        -76.695556,
        18.729501
      ],
      [
        -76.697034,
        18.732628
      ],
      [
        -76.698528,
        18.735781
      ],
      [
        -76.700035,
        18.738958
      ],
      [
        -76.701556,
        18.742161
      ],
      [
        -76.70309,
        18.745389
      ],
      [
        -76.704637,
        18.748641
      ],
      [
        -76.706196,
        18.751917
      ],
      [
        -76.707766,
```

(continues on next page)

(continued from previous page)

```
        18.755216
      ],
      [
        -76.709347,
        18.758539
      ],
      [
        -76.727297,
        18.79652
      ],
      [
        -76.728965,
        18.800094
      ],
      [
        -76.730637,
        18.803686
      ],
      [
        -76.732312,
        18.807297
      ],
      [
        -76.733989,
        18.810926
      ],
      [
        -76.735669,
        18.814573
      ],
      [
        -76.737349,
        18.818236
      ],
      [
        -76.739031,
        18.821917
      ],
      [
        -76.740713,
        18.825614
      ],
      [
        -76.742394,
        18.829327
      ],
      [
        -76.760714,
        18.871143
      ],
      [
        -76.762348,
```

(continues on next page)

(continued from previous page)

```
        18.875023
      ],
      [
        -76.763976,
        18.878915
      ],
      [
        -76.765595,
        18.882819
      ],
      [
        -76.767205,
        18.886733
      ],
      [
        -76.768807,
        18.890658
      ],
      [
        -76.770398,
        18.894593
      ],
      [
        -76.771979,
        18.898537
      ],
      [
        -76.77355,
        18.902492
      ],
      [
        -76.775108,
        18.906455
      ],
      [
        -76.791342,
        18.950559
      ],
      [
        -76.79272,
        18.954605
      ],
      [
        -76.79408,
        18.958656
      ],
      [
        -76.79542,
        18.962711
      ],
      [
        -76.79674,
```

(continues on next page)

(continued from previous page)

```
        18.966769
      ],
      [
        -76.79804,
        18.970832
      ],
      [
        -76.799319,
        18.974898
      ],
      [
        -76.800577,
        18.978966
      ],
      [
        -76.801813,
        18.983037
      ],
      [
        -76.803026,
        18.98711
      ],
      [
        -76.814718,
        19.031954
      ],
      [
        -76.815617,
        19.036026
      ],
      [
        -76.816485,
        19.040095
      ],
      [
        -76.817323,
        19.044161
      ],
      [
        -76.81813,
        19.048224
      ],
      [
        -76.818906,
        19.052283
      ],
      [
        -76.81965,
        19.056339
      ],
      [
        -76.820361,
```

(continues on next page)

(continued from previous page)

```
        19.06039
      ],
      [
        -76.821038,
        19.064437
      ],
      [
        -76.821682,
        19.068479
      ],
      [
        -76.826379,
        19.112518
      ],
      [
        -76.826575,
        19.116474
      ],
      [
        -76.826729,
        19.12042
      ],
      [
        -76.826842,
        19.124357
      ],
      [
        -76.826913,
        19.128284
      ],
      [
        -76.826941,
        19.1322
      ],
      [
        -76.826926,
        19.136105
      ],
      [
        -76.826867,
        19.139998
      ],
      [
        -76.826764,
        19.143881
      ],
      [
        -76.826615,
        19.147751
      ],
      [
        -76.821862,
```

(continues on next page)

(continued from previous page)

```
        19.189438
      ],
      [
        -76.821131,
        19.193137
      ],
      [
        -76.820348,
        19.196821
      ],
      [
        -76.819513,
        19.200487
      ],
      [
        -76.818624,
        19.204137
      ],
      [
        -76.817681,
        19.207769
      ],
      [
        -76.816684,
        19.211382
      ],
      [
        -76.815632,
        19.214978
      ],
      [
        -76.814525,
        19.218555
      ],
      [
        -76.813361,
        19.222113
      ],
      [
        -76.796702,
        19.259901
      ],
      [
        -76.794822,
        19.263204
      ],
      [
        -76.792879,
        19.266484
      ],
      [
        -76.790871,
```

(continues on next page)

(continued from previous page)

```
        19.269739
      ],
      [
        -76.7888,
        19.272971
      ],
      [
        -76.786663,
        19.276178
      ],
      [
        -76.784461,
        19.27936
      ],
      [
        -76.782193,
        19.282517
      ],
      [
        -76.779858,
        19.285648
      ],
      [
        -76.777456,
        19.288753
      ],
      [
        -76.746436,
        19.321095
      ],
      [
        -76.743184,
        19.323861
      ],
      [
        -76.739857,
        19.326597
      ],
      [
        -76.736455,
        19.329301
      ],
      [
        -76.732978,
        19.331974
      ],
      [
        -76.729424,
        19.334616
      ],
      [
        -76.725794,
```

(continues on next page)

(continued from previous page)

```
        19.337226
      ],
      [
        -76.722086,
        19.339803
      ],
      [
        -76.7183,
        19.342348
      ],
      [
        -76.714437,
        19.34486
      ],
      [
        -76.666601,
        19.370209
      ],
      [
        -76.661753,
        19.372297
      ],
      [
        -76.65682,
        19.374348
      ],
      [
        -76.6518,
        19.37636
      ],
      [
        -76.646693,
        19.378335
      ],
      [
        -76.641499,
        19.38027
      ],
      [
        -76.636218,
        19.382167
      ],
      [
        -76.630847,
        19.384024
      ],
      [
        -76.625388,
        19.385842
      ],
      [
        -76.61984,
```

(continues on next page)

(continued from previous page)

```
        19.38762
      ],
      [
        -76.552734,
        19.40443
      ],
      [
        -76.546083,
        19.405678
      ],
      [
        -76.539367,
        19.40684
      ],
      [
        -76.532587,
        19.407915
      ],
      [
        -76.525744,
        19.408906
      ],
      [
        -76.518837,
        19.409812
      ],
      [
        -76.511867,
        19.410635
      ],
      [
        -76.504836,
        19.411377
      ],
      [
        -76.497743,
        19.412037
      ],
      [
        -76.490589,
        19.412616
      ],
      [
        -76.408013,
        19.413952
      ],
      [
        -76.400167,
        19.413639
      ],
      [
        -76.392267,
```

(continues on next page)

(continued from previous page)

```
        19.413258
      ],
      [
        -76.384313,
        19.412809
      ],
      [
        -76.376306,
        19.412293
      ],
      [
        -76.368245,
        19.411713
      ],
      [
        -76.360133,
        19.411067
      ],
      [
        -76.351969,
        19.410357
      ],
      [
        -76.343754,
        19.409585
      ],
      [
        -76.335488,
        19.408751
      ],
      [
        -76.241373,
        19.395747
      ],
      [
        -76.23254,
        19.394241
      ],
      [
        -76.223663,
        19.392685
      ],
      [
        -76.214743,
        19.39108
      ],
      [
        -76.205781,
        19.389427
      ],
      [
        -76.196776,
```

(continues on next page)

(continued from previous page)

```
        19.387726
      ],
      [
        -76.187729,
        19.385979
      ],
      [
        -76.178641,
        19.384187
      ],
      [
        -76.169512,
        19.38235
      ],
      [
        -76.160343,
        19.38047
      ],
      [
        -76.056979,
        19.357173
      ],
      [
        -76.047368,
        19.354842
      ],
      [
        -76.037723,
        19.352479
      ],
      [
        -76.028046,
        19.350085
      ],
      [
        -76.018336,
        19.347662
      ],
      [
        -76.008594,
        19.345209
      ],
      [
        -75.998821,
        19.342729
      ],
      [
        -75.989017,
        19.340222
      ],
      [
        -75.979183,
```

(continues on next page)

(continued from previous page)

```
        19.337688
      ],
      [
        -75.969319,
        19.33513
      ],
      [
        -75.858998,
        19.305587
      ],
      [
        -75.848817,
        19.302798
      ],
      [
        -75.838613,
        19.299996
      ],
      [
        -75.828386,
        19.297181
      ],
      [
        -75.818138,
        19.294355
      ],
      [
        -75.807868,
        19.291519
      ],
      [
        -75.797577,
        19.288673
      ],
      [
        -75.787266,
        19.285819
      ],
      [
        -75.776935,
        19.282957
      ],
      [
        -75.766584,
        19.280088
      ],
      [
        -75.651598,
        19.248345
      ],
      [
        -75.641055,
```

(continues on next page)

(continued from previous page)

```
        19.245466
      ],
      [
        -75.6305,
        19.242593
      ],
      [
        -75.619933,
        19.239725
      ],
      [
        -75.609354,
        19.236865
      ],
      [
        -75.598764,
        19.234013
      ],
      [
        -75.588164,
        19.231169
      ],
      [
        -75.577553,
        19.228335
      ],
      [
        -75.566933,
        19.225512
      ],
      [
        -75.556304,
        19.222701
      ],
      [
        -75.438944,
        19.192804
      ],
      [
        -75.428249,
        19.190203
      ],
      [
        -75.417551,
        19.187627
      ],
      [
        -75.406851,
        19.185074
      ],
      [
        -75.39615,
```

(continues on next page)

(continued from previous page)

```
        19.182547
      ],
      [
        -75.385449,
        19.180047
      ],
      [
        -75.374747,
        19.177573
      ],
      [
        -75.364046,
        19.175128
      ],
      [
        -75.353346,
        19.172712
      ],
      [
        -75.342647,
        19.170326
      ],
      [
        -75.225205,
        19.146321
      ],
      [
        -75.214564,
        19.144367
      ],
      [
        -75.203932,
        19.142454
      ],
      [
        -75.193308,
        19.140584
      ],
      [
        -75.182694,
        19.138759
      ],
      [
        -75.172089,
        19.136978
      ],
      [
        -75.161495,
        19.135242
      ],
      [
        -75.150911,
```

(continues on next page)

(continued from previous page)

```
        19.133553
      ],
      [
        -75.140339,
        19.131912
      ],
      [
        -75.129778,
        19.130319
      ],
      [
        -75.014547,
        19.116253
      ],
      [
        -75.004169,
        19.115312
      ],
      [
        -74.993811,
        19.114432
      ],
      [
        -74.983471,
        19.113613
      ],
      [
        -74.973152,
        19.112857
      ],
      [
        -74.962852,
        19.112163
      ],
      [
        -74.952573,
        19.111533
      ],
      [
        -74.942315,
        19.110969
      ],
      [
        -74.932079,
        19.11047
      ],
      [
        -74.921866,
        19.110038
      ],
      [
        -74.811136,
```

(continues on next page)

(continued from previous page)

```
        19.109957
      ],
      [
        -74.801231,
        19.110398
      ],
      [
        -74.791354,
        19.110918
      ],
      [
        -74.781507,
        19.111517
      ],
      [
        -74.77169,
        19.112197
      ],
      [
        -74.761904,
        19.112959
      ],
      [
        -74.752149,
        19.113803
      ],
      [
        -74.742426,
        19.114731
      ],
      [
        -74.732735,
        19.115742
      ],
      [
        -74.723076,
        19.116839
      ],
      [
        -74.61914,
        19.134789
      ],
      [
        -74.609938,
        19.136963
      ],
      [
        -74.600822,
        19.139201
      ],
      [
        -74.591791,
```

(continues on next page)

(continued from previous page)

```
        19.141502
      ],
      [
        -74.582845,
        19.143866
      ],
      [
        -74.573982,
        19.146291
      ],
      [
        -74.565203,
        19.148777
      ],
      [
        -74.556507,
        19.151324
      ],
      [
        -74.547892,
        19.153931
      ],
      [
        -74.539359,
        19.156598
      ],
      [
        -74.450684,
        19.189694
      ],
      [
        -74.443078,
        19.193029
      ],
      [
        -74.435545,
        19.196417
      ],
      [
        -74.428085,
        19.199856
      ],
      [
        -74.420696,
        19.203347
      ],
      [
        -74.413378,
        19.206887
      ],
      [
        -74.406131,
```

(continues on next page)

(continued from previous page)

```
        19.210477
      ],
      [
        -74.398954,
        19.214116
      ],
      [
        -74.391847,
        19.217804
      ],
      [
        -74.384808,
        19.221539
      ],
      [
        -74.311729,
        19.265633
      ],
      [
        -74.305465,
        19.2699
      ],
      [
        -74.299261,
        19.274206
      ],
      [
        -74.293116,
        19.278553
      ],
      [
        -74.287031,
        19.282939
      ],
      [
        -74.281005,
        19.287364
      ],
      [
        -74.275036,
        19.291827
      ],
      [
        -74.269124,
        19.296327
      ],
      [
        -74.263269,
        19.300865
      ],
      [
        -74.25747,
```

(continues on next page)

(continued from previous page)

```
        19.305439
      ],
      [
        -74.197194,
        19.357992
      ],
      [
        -74.192017,
        19.362958
      ],
      [
        -74.186888,
        19.367953
      ],
      [
        -74.181805,
        19.372977
      ],
      [
        -74.17677,
        19.378028
      ],
      [
        -74.17178,
        19.383107
      ],
      [
        -74.166835,
        19.388212
      ],
      [
        -74.161935,
        19.393343
      ],
      [
        -74.157079,
        19.398499
      ],
      [
        -74.152266,
        19.403681
      ],
      [
        -74.101996,
        19.462155
      ],
      [
        -74.097652,
        19.467591
      ],
      [
        -74.093344,
```

(continues on next page)

(continued from previous page)

```
        19.473043
      ],
      [
        -74.08907,
        19.478513
      ],
      [
        -74.08483,
        19.483998
      ],
      [
        -74.080623,
        19.4895
      ],
      [
        -74.076448,
        19.495016
      ],
      [
        -74.072305,
        19.500547
      ],
      [
        -74.068193,
        19.506092
      ],
      [
        -74.064112,
        19.51165
      ],
      [
        -74.021055,
        19.573508
      ],
      [
        -74.017291,
        19.579181
      ],
      [
        -74.013549,
        19.58486
      ],
      [
        -74.009829,
        19.590545
      ],
      [
        -74.00613,
        19.596234
      ],
      [
        -74.002451,
```

(continues on next page)

(continued from previous page)

```
        19.601928
      ],
      [
        -73.998792,
        19.607625
      ],
      [
        -73.995153,
        19.613325
      ],
      [
        -73.991532,
        19.619027
      ],
      [
        -73.987928,
        19.624732
      ],
      [
        -73.949288,
        19.687433
      ],
      [
        -73.945849,
        19.693115
      ],
      [
        -73.94242,
        19.69879
      ],
      [
        -73.939,
        19.704459
      ],
      [
        -73.935589,
        19.710121
      ],
      [
        -73.932185,
        19.715776
      ],
      [
        -73.928788,
        19.721423
      ],
      [
        -73.925397,
        19.727061
      ],
      [
        -73.922012,
```

(continues on next page)

(continued from previous page)

```
        19.732691
      ],
      [
        -73.918633,
        19.73831
      ],
      [
        -73.881615,
        19.799318
      ],
      [
        -73.878248,
        19.804776
      ],
      [
        -73.874877,
        19.810216
      ],
      [
        -73.871503,
        19.815639
      ],
      [
        -73.868125,
        19.821044
      ],
      [
        -73.864741,
        19.826429
      ],
      [
        -73.861352,
        19.831795
      ],
      [
        -73.857956,
        19.837141
      ],
      [
        -73.854554,
        19.842466
      ],
      [
        -73.851144,
        19.84777
      ],
      [
        -73.812954,
        19.904546
      ],
      [
        -73.809404,
```

(continues on next page)

(continued from previous page)

```
        19.90955
      ],
      [
        -73.805838,
        19.914525
      ],
      [
        -73.802256,
        19.91947
      ],
      [
        -73.798656,
        19.924386
      ],
      [
        -73.795039,
        19.929272
      ],
      [
        -73.791404,
        19.934126
      ],
      [
        -73.787749,
        19.938949
      ],
      [
        -73.784075,
        19.943739
      ],
      [
        -73.780381,
        19.948497
      ],
      [
        -73.738224,
        19.998501
      ],
      [
        -73.734237,
        20.002821
      ],
      [
        -73.730222,
        20.007099
      ],
      [
        -73.726177,
        20.011337
      ],
      [
        -73.722103,
```

(continues on next page)

(continued from previous page)

```
        20.015534
      ],
      [
        -73.717998,
        20.019688
      ],
      [
        -73.713862,
        20.0238
      ],
      [
        -73.709694,
        20.027869
      ],
      [
        -73.705494,
        20.031894
      ],
      [
        -73.701261,
        20.035874
      ],
      [
        -73.652343,
        20.07657
      ],
      [
        -73.647695,
        20.079975
      ],
      [
        -73.643066,
        20.083332
      ],
      [
        -73.638456,
        20.086641
      ],
      [
        -73.633865,
        20.089903
      ],
      [
        -73.629294,
        20.093117
      ],
      [
        -73.624741,
        20.096285
      ],
      [
        -73.620208,
```

(continues on next page)

(continued from previous page)

```
        20.099406
      ],
      [
        -73.615694,
        20.10248
      ],
      [
        -73.6112,
        20.105509
      ],
      [
        -73.563048,
        20.135879
      ],
      [
        -73.558789,
        20.138381
      ],
      [
        -73.55455,
        20.140841
      ],
      [
        -73.550331,
        20.143258
      ],
      [
        -73.546131,
        20.145635
      ],
      [
        -73.541952,
        20.147971
      ],
      [
        -73.537793,
        20.150266
      ],
      [
        -73.533654,
        20.152521
      ],
      [
        -73.529535,
        20.154736
      ],
      [
        -73.525436,
        20.156911
      ],
      [
        -73.481695,
```

(continues on next page)

(continued from previous page)

```
        20.178306
      ],
      [
        -73.477841,
        20.180029
      ],
      [
        -73.474009,
        20.181716
      ],
      [
        -73.470198,
        20.183368
      ],
      [
        -73.466407,
        20.184985
      ],
      [
        -73.462637,
        20.186568
      ],
      [
        -73.458888,
        20.188116
      ],
      [
        -73.45516,
        20.18963
      ],
      [
        -73.451453,
        20.19111
      ],
      [
        -73.447767,
        20.192557
      ],
      [
        -73.408625,
        20.206355
      ],
      [
        -73.405195,
        20.207425
      ],
      [
        -73.401787,
        20.208465
      ],
      [
        -73.3984,
```

(continues on next page)

(continued from previous page)

```
        20.209477
      ],
      [
        -73.395035,
        20.21046
      ],
      [
        -73.391692,
        20.211414
      ],
      [
        -73.388371,
        20.212341
      ],
      [
        -73.385071,
        20.213239
      ],
      [
        -73.381794,
        20.21411
      ],
      [
        -73.378538,
        20.214954
      ],
      [
        -73.344183,
        20.222533
      ],
      [
        -73.341194,
        20.223075
      ],
      [
        -73.338227,
        20.223594
      ],
      [
        -73.335282,
        20.224091
      ],
      [
        -73.33236,
        20.224565
      ],
      [
        -73.329461,
        20.225016
      ],
      [
        -73.326584,
```

(continues on next page)

(continued from previous page)

```
        20.225446
      ],
      [
        -73.32373,
        20.225854
      ],
      [
        -73.320899,
        20.226242
      ],
      [
        -73.318091,
        20.226608
      ],
      [
        -73.288713,
        20.229346
      ],
      [
        -73.286181,
        20.229486
      ],
      [
        -73.283672,
        20.229609
      ],
      [
        -73.281187,
        20.229715
      ],
      [
        -73.278725,
        20.229806
      ],
      [
        -73.276287,
        20.22988
      ],
      [
        -73.273872,
        20.229939
      ],
      [
        -73.271481,
        20.229982
      ],
      [
        -73.269113,
        20.230011
      ],
      [
        -73.266769,
```

(continues on next page)

(continued from previous page)

```
        20.230025
      ],
      [
        -73.242557,
        20.229301
      ],
      [
        -73.2405,
        20.229163
      ],
      [
        -73.238467,
        20.229016
      ],
      [
        -73.236459,
        20.228857
      ],
      [
        -73.234474,
        20.228689
      ],
      [
        -73.232514,
        20.228511
      ],
      [
        -73.230578,
        20.228324
      ],
      [
        -73.228667,
        20.228128
      ],
      [
        -73.22678,
        20.227924
      ],
      [
        -73.224918,
        20.22771
      ],
      [
        -73.20606,
        20.224903
      ],
      [
        -73.204495,
        20.224614
      ],
      [
        -73.202955,
```

(continues on next page)

(continued from previous page)

```
        20.22432
      ],
      [
        -73.20144,
        20.224023
      ],
      [
        -73.19995,
        20.223722
      ],
      [
        -73.198486,
        20.223417
      ],
      [
        -73.197046,
        20.22311
      ],
      [
        -73.195632,
        20.222799
      ],
      [
        -73.194243,
        20.222486
      ],
      [
        -73.19288,
        20.222172
      ],
      [
        -73.179566,
        20.218659
      ],
      [
        -73.17851,
        20.218343
      ],
      [
        -73.17748,
        20.218029
      ],
      [
        -73.176476,
        20.217718
      ],
      [
        -73.175498,
        20.217409
      ],
      [
        -73.174545,
```

(continues on next page)

(continued from previous page)

```
        20.217103
      ],
      [
        -73.173619,
        20.2168
      ],
      [
        -73.172719,
        20.216501
      ],
      [
        -73.171846,
        20.216205
      ],
      [
        -73.170998,
        20.215914
      ],
      [
        -73.163417,
        20.213074
      ],
      [
        -73.162888,
        20.212857
      ],
      [
        -73.162385,
        20.212649
      ],
      [
        -73.161909,
        20.212449
      ],
      [
        -73.161459,
        20.212257
      ],
      [
        -73.161037,
        20.212075
      ],
      [
        -73.160641,
        20.211902
      ],
      [
        -73.160273,
        20.21174
      ],
      [
        -73.159932,
```

(continues on next page)

(continued from previous page)

```
        20.211587
      ],
      [
        -73.159617,
        20.211444
      ]
    ],
    "type": "LineString"
  },
  "properties": {},
  "type": "Feature"
}
```

25.2 Interactive Example

```
from geojson import LineString, Feature
from ipyleaflet import Map, GeoJSON
from turfpy.transformation import bezier_spline

ls = LineString([(-76.091308, 18.427501),
                 (-76.695556, 18.729501),
                 (-76.552734, 19.40443),
                 (-74.61914, 19.134789),
                 (-73.652343, 20.07657),
                 (-73.157958, 20.210656)])
f = Feature(geometry=ls)


geo_json = GeoJSON(data=f)

spline_geo_json = GeoJSON(data=bezier_spline(f), style={'color': 'red'})

m = Map(center=[19.318170089962457, -74.85710620880128], zoom=8)

m.add_layer(geo_json)
m.add_layer(spline_geo_json)

m
```

CHAPTER

TWENTYSIX

CONCAVE HULL

Generate concave hull for the given feature or Feature Collection.

26.1 Example

```
import json
from turfpy.transformation import concave
from geojson import FeatureCollection, Feature, Point
f1 = Feature(geometry=Point((-63.601226, 44.642643)))
f2 = Feature(geometry=Point((-63.591442, 44.651436)))
f3 = Feature(geometry=Point((-63.580799, 44.648749)))
f4 = Feature(geometry=Point((-63.573589, 44.641788)))
f5 = Feature(geometry=Point((-63.587665, 44.64533)))
f6 = Feature(geometry=Point((-63.595218, 44.64765)))
fc = [f1, f2, f3, f4, f5, f6]
ch = concave(FeatureCollection(fc), alpha=100)
print(json.dumps(ch, indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      [
        [
          -63.587665,
          44.64533
        ],
        [
          -63.601226,
          44.642643
        ],
        [
          -63.595218,
          44.64765
        ],
        [
          -63.591442,
          44.651436
        ],
        [
```

(continues on next page)

(continued from previous page)

```
          -63.580799,
          44.648749
        ],
        [
          -63.573589,
          44.641788
        ],
        [
          -63.587665,
          44.64533
        ]
      ]
    ],
    "type": "Polygon"
  },
  "properties": {},
  "type": "Feature"
}
```

26.2 Interactive Example

```
from geojson import FeatureCollection, Feature, Point
from ipyleaflet import Map, GeoJSON
from turfpy.transformation import concave

f1 = Feature(geometry=Point((-63.601226, 44.642643)))
f2 = Feature(geometry=Point((-63.591442, 44.651436)))
f3 = Feature(geometry=Point((-63.580799, 44.648749)))
f4 = Feature(geometry=Point((-63.573589, 44.641788)))
f5 = Feature(geometry=Point((-63.587665, 44.64533)))
f6 = Feature(geometry=Point((-63.595218, 44.64765)))
fc = [f1, f2, f3, f4, f5, f6]
fc = FeatureCollection(fc)

geo_json = GeoJSON(data=fc)

spline_geo_json = GeoJSON(data=concave(FeatureCollection(fc), alpha=100), style={'color
↪': 'red'})

m = Map(center=[44.64740465397292, -63.58361206948757], zoom=14)

m.add_layer(geo_json)
m.add_layer(spline_geo_json)

m
```

CHAPTER

TWENTYSEVEN

CONVEX HULL

27.1 Example

```
import json
from turfpy.transformation import convex
from geojson import FeatureCollection, Feature, Point
f1 = Feature(geometry=Point((-63.601226, 44.642643)))
f2 = Feature(geometry=Point((-63.591442, 44.651436)))
f3 = Feature(geometry=Point((-63.580799, 44.648749)))
f4 = Feature(geometry=Point((-63.573589, 44.641788)))
f5 = Feature(geometry=Point((-63.587665, 44.64533)))
f6 = Feature(geometry=Point((-63.595218, 44.64765)))
fc = [f1, f2, f3, f4, f5, f6]
ch = convex(FeatureCollection(fc))
print(json.dumps(ch, indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      [
        [
          -63.573589,
          44.641788
        ],
        [
          -63.601226,
          44.642643
        ],
        [
          -63.591442,
          44.651436
        ],
        [
          -63.580799,
          44.648749
        ],
        [
          -63.573589,
          44.641788
        ]
```

(continues on next page)

(continued from previous page)

```
      ]
    ],
    "type": "Polygon"
  },
  "properties": {},
  "type": "Feature"
}
```

27.2 Interactive Example

```
from geojson import FeatureCollection, Feature, Point
from ipyleaflet import Map, GeoJSON
from turfpy.transformation import convex

f1 = Feature(geometry=Point((-63.601226, 44.642643)))
f2 = Feature(geometry=Point((-63.591442, 44.651436)))
f3 = Feature(geometry=Point((-63.580799, 44.648749)))
f4 = Feature(geometry=Point((-63.573589, 44.641788)))
f5 = Feature(geometry=Point((-63.587665, 44.64533)))
f6 = Feature(geometry=Point((-63.595218, 44.64765)))
fc = [f1, f2, f3, f4, f5, f6]
fc = FeatureCollection(fc)

geo_json = GeoJSON(data=fc)

spline_geo_json = GeoJSON(data=convex(FeatureCollection(fc)), style={'color': 'red'})

m = Map(center=[44.64740465397292, -63.58361206948757], zoom=14)

m.add_layer(geo_json)
m.add_layer(spline_geo_json)

m
```

CHAPTER

TWENTYEIGHT

INTERSECT

Takes polygons and finds their intersection.

28.1 Example

```
import json
from turfpy.transformation import intersect
from geojson import Feature
f = Feature(geometry={"coordinates": [
[[-122.801742, 45.48565], [-122.801742, 45.60491],
[-122.584762, 45.60491], [-122.584762, 45.48565],
[-122.801742, 45.48565]]], "type": "Polygon"})
b = Feature(geometry={"coordinates": [
[[-122.520217, 45.535693], [-122.64038, 45.553967],
[-122.720031, 45.526554], [-122.669906, 45.507309],
[-122.723464, 45.446643], [-122.532577, 45.408574],
[-122.487258, 45.477466], [-122.520217, 45.535693]
]], "type": "Polygon"})
it = intersect([f, b])
print(json.dumps(it, indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      [
        [
          -122.689027,
          45.48565
        ],
        [
          -122.669906,
          45.507309
        ],
        [
          -122.720031,
          45.526554
        ],
        [
          -122.64038,
```

(continues on next page)

(continued from previous page)

```
          45.553967
        ],
        [
          -122.584762,
          45.545509
        ],
        [
          -122.584762,
          45.48565
        ],
        [
          -122.689027,
          45.48565
        ]
      ]
    ],
    "type": "Polygon"
  },
  "properties": {},
  "type": "Feature"
}
```

28.2 Interactive Example

```
from geojson import Feature
from ipyleaflet import Map, GeoJSON, LayersControl
from turfpy.transformation import intersect

f = Feature(geometry={"coordinates": [
[[-122.801742, 45.48565], [-122.801742, 45.60491],
[-122.584762, 45.60491], [-122.584762, 45.48565],
[-122.801742, 45.48565]]], "type": "Polygon"})
b = Feature(geometry={"coordinates": [
[[-122.520217, 45.535693], [-122.64038, 45.553967],
[-122.720031, 45.526554], [-122.669906, 45.507309],
[-122.723464, 45.446643], [-122.532577, 45.408574],
[-122.487258, 45.477466], [-122.520217, 45.535693]
]], "type": "Polygon"})

geo_json_1 = GeoJSON(name="First Polygon", data=f)

geo_json_2 = GeoJSON(name='Second Polygon', data=b, style={'color': 'green'})

geojson = GeoJSON(name='Intersection', data=intersect([f, b]), style={'color': 'red'})


m = Map(center=[45.510343157077976, -122.63075172901155], zoom=10)

m.add_layer(geo_json_1)
m.add_layer(geo_json_2)
```

(continues on next page)

(continued from previous page)

```
m.add_layer(geojson)

control = LayersControl(position='topright')
m.add_control(control)

m
```

CHAPTER

TWENTYNINE

UNION

Given list of features or FeatureCollection return union of those.

29.1 Example

```
import json
from turfpy.transformation import union
from geojson import Feature, Polygon, FeatureCollection
f1 = Feature(geometry=Polygon([[
    [-82.574787, 35.594087],
    [-82.574787, 35.615581],
    [-82.545261, 35.615581],
    [-82.545261, 35.594087],
     [-82.574787, 35.594087]
]]), properties={"fill": "#00f"})
f2 = Feature(geometry=Polygon([[
    [-82.560024, 35.585153],
    [-82.560024, 35.602602],
    [-82.52964, 35.602602],
    [-82.52964, 35.585153],
    [-82.560024, 35.585153]]]), properties={"fill": "#00f"})
un = union(FeatureCollection([f1, f2], properties={"combine": "yes"}))
print(json.dumps(un, indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      [
        [
          -82.560024,
          35.585153
        ],
        [
          -82.560024,
          35.594087
        ],
        [
          -82.574787,
          35.594087
```

(continues on next page)

(continued from previous page)

```
        ],
        [
          -82.574787,
          35.615581
        ],
        [
          -82.545261,
          35.615581
        ],
        [
          -82.545261,
          35.602602
        ],
        [
          -82.52964,
          35.602602
        ],
        [
          -82.52964,
          35.585153
        ],
        [
          -82.560024,
          35.585153
        ]
      ]
    ],
    "type": "Polygon"
  },
  "properties": {
    "combine": "yes",
    "fill": "#00f"
  },
  "type": "Feature"
}
```

29.2 Interactive Example

```
from geojson import Feature, Polygon, FeatureCollection
from ipyleaflet import Map, GeoJSON, LayersControl
from turfpy.transformation import union

f1 = Feature(
    geometry=Polygon(
        [
            [
                [-82.574787, 35.594087],
                [-82.574787, 35.615581],
                [-82.545261, 35.615581],
                [-82.545261, 35.594087],
```

(continues on next page)

(continued from previous page)

```
                [-82.574787, 35.594087],
            ]
        ]
    ),
    properties={"fill": "#00f"},
)
f2 = Feature(
    geometry=Polygon(
        [
            [
                [-82.560024, 35.585153],
                [-82.560024, 35.602602],
                [-82.52964, 35.602602],
                [-82.52964, 35.585153],
                [-82.560024, 35.585153],
            ]
        ]
    ),
    properties={"fill": "#00f"},
)

geo_json_1 = GeoJSON(name="First Polygon", data=f1)

geo_json_2 = GeoJSON(name="Second Polygon", data=f2, style={"color": "green"})

geojson = GeoJSON(
    name="Union",
    data=union(FeatureCollection([f1, f2], properties={"combine": "yes"})),
    style={"color": "red"},
)

m = Map(center=[35.60069336198429, -82.548927962780001], zoom=13)
```

CHAPTER

THIRTY

DISSOLVE

Take FeatureCollection or list of features to dissolve based on property_name provided.

30.1 Example

```
import json
from geojson import Polygon, Feature, FeatureCollection
from turfpy.transformation import dissolve
f1 = Feature(geometry=Polygon([[
    [0, 0],
    [0, 1],
    [1, 1],
    [1, 0],
    [0, 0]]]), properties={"combine": "yes", "fill": "#00f"})
f2 = Feature(geometry=Polygon([[
    [0, -1],
    [0, 0],
    [1, 0],
    [1, -1],
    [0,-1]]]), properties={"combine": "yes"})
f3 = Feature(geometry=Polygon([[
    [1,-1],
    [1, 0],
    [2, 0],
    [2, -1],
    [1, -1]]]), properties={"combine": "no"})
ds = dissolve(FeatureCollection([f1, f2, f3]), property_name='combine')
print(json.dumps(ds, indent=2, sort_keys=True))
```

```
{
  "features": [
    {
      "geometry": {
        "coordinates": [
          [
            [
              0.0,
              -1.0
            ],
```

(continues on next page)

(continued from previous page)

```
            [
              0.0,
              0.0
            ],
            [
              0.0,
              1.0
            ],
            [
              1.0,
              1.0
            ],
            [
              1.0,
              0.0
            ],
            [
              1.0,
              -1.0
            ],
            [
              0.0,
              -1.0
            ]
          ]
        ],
        "type": "Polygon"
      },
      "properties": {
        "combine": "yes",
        "fill": "#00f"
      },
      "type": "Feature"
    },
    {
      "geometry": {
        "coordinates": [
          [
            [
              1.0,
              -1.0
            ],
            [
              1.0,
              0.0
            ],
            [
              2.0,
              0.0
            ],
            [
              2.0,
```

(continues on next page)

(continued from previous page)

```
              -1.0
            ],
            [
              1.0,
              -1.0
            ]
          ]
        ],
        "type": "Polygon"
      },
      "properties": {
        "combine": "no"
      },
      "type": "Feature"
    }
  ],
  "type": "FeatureCollection"
}
```

30.2 Interactive Example

```
from geojson import Feature, Polygon, FeatureCollection
from ipyleaflet import Map, GeoJSON, LayersControl
from turfpy.transformation import dissolve

f1 = Feature(
    geometry=Polygon([[[0, 0], [0, 1], [1, 1], [1, 0], [0, 0]]]),
    properties={"combine": "yes", "fill": "#00f"},
)
f2 = Feature(
    geometry=Polygon([[[0, -1], [0, 0], [1, 0], [1, -1], [0, -1]]]),
    properties={"combine": "yes"},
)
f3 = Feature(
    geometry=Polygon([[[1, -1], [1, 0], [2, 0], [2, -1], [1, -1]]]),
    properties={"combine": "no"},
)

geo_json_1 = GeoJSON(name="First Polygon", data=f1)

geo_json_2 = GeoJSON(name="Second Polygon", data=f2, style={"color": "green"})

geo_json_3 = GeoJSON(name="Third Polygon", data=f3, style={"color": "black"})

geojson = GeoJSON(
    name="Dissolve",
    data=dissolve(FeatureCollection([f1, f2, f3]), property_name="combine"),
    style={"color": "red"},
)
```

(continues on next page)

(continued from previous page)

```
m = Map(center=[0.257748688144287, 1.9686126708984377], zoom=7)


m.add_layer(geo_json_1)
m.add_layer(geo_json_2)
m.add_layer(geo_json_3)
m.add_layer(geojson)

control = LayersControl(position="topright")
m.add_control(control)

m
```

CHAPTER

THIRTYONE

DIFFERENCE

Find the difference between given two features.

31.1 Example

```
import json
from geojson import Polygon, Feature
from turfpy.transformation import difference
f1 = Feature(geometry=Polygon([[
    [128, -26],
    [141, -26],
    [141, -21],
    [128, -21],
    [128, -26]]]), properties={"combine": "yes", "fill": "#00f"})
f2 = Feature(geometry=Polygon([[
    [126, -28],
    [140, -28],
    [140, -20],
    [126, -20],
    [126, -28]]]), properties={"combine": "yes"})
diff = difference(f1, f2)
print(json.dumps(diff, indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      [
        [
          140.0,
          -21.0
        ],
        [
          141.0,
          -21.0
        ],
        [
          141.0,
          -26.0
        ],
```

(continues on next page)

(continued from previous page)

```
        [
          140.0,
          -26.0
        ],
        [
          140.0,
          -21.0
        ]
      ]
    ],
    "type": "Polygon"
  },
  "properties": {
    "combine": "yes",
    "fill": "#00f"
  },
  "type": "Feature"
}
```

31.2 Interactive Example

```
from geojson import Feature, Polygon, FeatureCollection
from ipyleaflet import Map, GeoJSON, LayersControl
from turfpy.transformation import difference

f1 = Feature(geometry=Polygon([[
    [128, -26],
    [141, -26],
    [141, -21],
    [128, -21],
    [128, -26]]]), properties={"combine": "yes", "fill": "#00f"})
f2 = Feature(geometry=Polygon([[
    [126, -28],
    [140, -28],
    [140, -20],
    [126, -20],
    [126, -28]]]), properties={"combine": "yes"})

geo_json_1 = GeoJSON(name="First Polygon", data=f1)

geo_json_2 = GeoJSON(name='Second Polygon', data=f2, style={'color': 'green'})

geojson = GeoJSON(name='Difference', data=difference(f1, f2), style={'color': 'red'})


m = Map(center=[-22.71491497943416, 135.750846862793], zoom=5)


m.add_layer(geo_json_1)
m.add_layer(geo_json_2)
```

(continues on next page)

(continued from previous page)

```
m.add_layer(geojson)

control = LayersControl(position='topright')
m.add_control(control)

m
```

CHAPTER

THIRTYTWO

TRANSFORM ROTATE

Rotates any geojson Feature or Geometry of a specified angle, around its centroid or a given pivot point; all rotations follow the right-hand rule.

32.1 Example

```
import json
from turfpy.transformation import transform_rotate
from geojson import Polygon, Feature
f = Feature(geometry=Polygon([[[0,29],[3.5,29],[2.5,32],[0,29]]]))
pivot = [0, 25]
tr = transform_rotate(f, 10, pivot)
print(json.dumps(tr, indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      [
        [
          0.779582,
          28.939231
        ],
        [
          4.215029,
          28.397872
        ],
        [
          3.837175,
          31.512519
        ],
        [
          0.779582,
          28.939231
        ]
      ]
    ],
    "type": "Polygon"
  },
  "properties": {},
```

(continues on next page)

(continued from previous page)

```
  "type": "Feature"
}
```

32.2 Interactive Example

```
from ipyleaflet import Map, GeoJSON, LayersControl

original = GeoJSON(name='Original', data=f)

rotated = GeoJSON(name='Rotated', data=transform_rotate(f, 10, pivot), style={'color':
↪'red'})

m = Map(center=[30.18519925274955, 2.939529418945313], zoom=5)

m.add_layer(original)
m.add_layer(rotated)

control = LayersControl(position='topright')
m.add_control(control)
m
```

CHAPTER

THIRTYTHREE

TRANSFORM TRANSLATE

Moves any geojson Feature or Geometry of a specified distance along a Rhumb Line on the provided direction angle.

33.1 Example

```
import json
from turfpy.transformation import transform_translate
from geojson import Polygon, Feature
f = Feature(geometry=Polygon([[[0,29],[3.5,29],[2.5,32],[0,29]]]))
tt = transform_translate(f, 100, direction=35, mutate=True)
print(json.dumps(tt, indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      [
        [
          0.591903,
          29.73668
        ],
        [
          4.091903,
          29.73668
        ],
        [
          3.110728,
          32.73668
        ],
        [
          0.591903,
          29.73668
        ]
      ]
    ],
    "type": "Polygon"
  },
  "properties": {},
  "type": "Feature"
}
```

33.2 Interactive Example

```
from ipyleaflet import Map, GeoJSON, LayersControl

original = GeoJSON(name='Original', data=f)

rotated = GeoJSON(name='Translated', data=transform_translate(f, 100, 35, mutate=True),␣
↪style={'color': 'red'})

m = Map(center=[30.18519925274955, 2.939529418945313], zoom=5)

m.add_layer(original)
m.add_layer(rotated)

control = LayersControl(position='topright')
m.add_control(control)
m
```

CHAPTER

THIRTYFOUR

TRANSFORM SCALE

Scale a GeoJSON from a given point by a factor of scaling (ex: factor=2 would make the GeoJSON 200% larger). If a FeatureCollection is provided, the origin point will be calculated based on each individual Feature.

34.1 Example

```
import json
from turfpy.transformation import transform_scale
from geojson import Polygon, Feature
f = Feature(geometry=Polygon([[[0,29],[3.5,29],[2.5,32],[0,29]]]))
ts = transform_scale(f, 3, origin=[0, 29])
print(json.dumps(ts, indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      [
        [
          0.0,
          29.0
        ],
        [
          10.5,
          29.0
        ],
        [
          7.763024,
          38.0
        ],
        [
          0.0,
          29.0
        ]
      ]
    ],
    "type": "Polygon"
  },
  "properties": {},
```

(continues on next page)

(continued from previous page)

```
  "type": "Feature"
}
```

34.2 Interactive Example

```
from ipyleaflet import Map, GeoJSON, LayersControl

original = GeoJSON(name='Original', data=f)

rotated = GeoJSON(name='Scaled', data=transform_scale(f, 3, origin=[0, 29]), style={
↪'color': 'red'})

m = Map(center=[33.52608402076209, 7.55413055419922], zoom=5)

m.add_layer(original)
m.add_layer(rotated)

control = LayersControl(position='topright')
m.add_control(control)
m
```

CHAPTER

THIRTYFIVE

TESSELATE

Tesselates a Feature into a FeatureCollection of triangles using earcut.

35.1 Interactive Example

```
from geojson import Feature
from turfpy.transformation import tesselate

f = Feature(
    geometry={
        "coordinates": [
            [[11, 0], [22, 4], [31, 0], [31, 11], [21, 15], [11, 11], [11, 0]]
        ],
        "type": "Polygon",
    }
)

from ipyleaflet import Map, GeoJSON, LayersControl

m = Map(center=(4.595931675360621, 29.52129364013672), zoom=4)
geo_json = GeoJSON(
    name="original",
    data=dict(f),
    style={"opacity": 1, "fillOpacity": 0.3, "weight": 1},
    hover_style={"color": "green", "dashArray": "0", "fillOpacity": 0.5},
)

result = tesselate(f)
m = Map(center=(4.595931675360621, 29.52129364013672), zoom=4)
geo_json2 = GeoJSON(
    name="tesselate",
    data=result,
    style={"opacity": 1, "fillOpacity": 0.3, "weight": 1},
    hover_style={"color": "green", "dashArray": "0", "fillOpacity": 0.5},
)
m.add_layer(geo_json2)
control = LayersControl(position="topright")
m.add_control(control)
m.add_layer(geo_json)
m
```

CHAPTER

THIRTYSIX

LINE OFFSET

Takes a linestring or multilinestring and returns a line at offset by the specified distance.

36.1 Example

```
import json
from geojson import MultiLineString, Feature
from turfpy.transformation import line_offset
ls = Feature(geometry=MultiLineString([
     [(3.75, 9.25), (-130.95, 1.52)],
     [(23.15, -34.25), (-1.35, -4.65), (3.45, 77.95)]
 ]))
lo = line_offset(ls, 2, unit='mi')
print(json.dumps(lo, indent=2, sort_keys=True))
```

```
{
  "geometry": {
    "coordinates": [
      [
        [
          3.748342,
          9.278899
        ],
        [
          -130.951658,
          1.548899
        ]
      ],
      [
        [
          23.172299,
          -34.231543
        ],
        [
          -1.320442,
          -4.640314
        ],
        [
          3.478898,
```

(continues on next page)

(continued from previous page)

```
          77.948321
        ]
      ]
    ],
    "type": "MultiLineString"
  },
  "properties": {},
  "type": "Feature"
}
```

36.2 Interactive Example

```
from ipyleaflet import Map, GeoJSON, LayersControl

original = GeoJSON(name='Original', data=ls)

rotated = GeoJSON(name='Offset Line', data=line_offset(ls, 2, unit='mi'), style={'color
↪': 'red'})

m = Map(center=[33.54139466898275, 7.5366210937500001], zoom=1)

m.add_layer(original)
m.add_layer(rotated)

control = LayersControl(position='topright')
m.add_control(control)
m
```

CHAPTER

THIRTYSEVEN

VORONOI

Takes a FeatureCollection of points, and a bounding box, and returns a FeatureCollection of Voronoi polygons.

37.1 Interactive Example

```
from turfpy.transformation import voronoi
from ipyleaflet import Map, GeoJSON, LayersControl, Marker

points = [
    [-66.97035175371198, 40.318345269995234],
    [-63.776304623759685, 40.45001590163247],
    [-65.41960483851452, 42.13985310302137],
    [-69.58133775268497, 43.95405461286195],
    [-65.66337553550034, 55.97088945355232],
    [-60.280418548905, 56.240669185466146],
    [-68.5129561347689, 50.12984589640148],
    [-64.2393519226657, 59.66235385923687],
    [-60.90340080176197, 47.319854339485836],
    [-64.013392235184, 47.48505628754743],
    [-63.5901237814622, 56.435271304519375],
    [-65.29273842230272, 40.46532951990474],
    [-64.34074762923518, 44.41467903845307],
    [-62.75238118105902, 50.232555998274044],
    [-66.39216606663494, 42.05774847944996],
    [-69.4747201276131, 43.288490094187104],
    [-60.22111554327566, 53.23070074089705],
    [-68.38456573407701, 58.12528229586756],
    [-60.022015858921954, 57.16751878858943],
    [-67.27241103543064, 46.52814784233641],
    [-68.44640005497628, 49.31604545534581],
    [-63.19928242836677, 40.397501696572654],
    [-69.75867935840617, 45.47856756289996],
    [-61.32766380696367, 58.672721168560955],
    [-67.52671607410451, 55.62959559068972],
    [-64.05030735753337, 56.19827268873002],
    [-65.20026060312107, 52.91890866887687],
    [-63.02174849203104, 45.34033157319018],
    [-62.33913739166953, 43.46950736683852],
    [-68.70183827840249, 51.16845764645581],
```

(continues on next page)

(continued from previous page)

```
    [-66.20557198738521, 45.61818662858589],
    [-67.06923935761512, 56.49552328077417],
    [-68.45745370027925, 53.01501887198211],
    [-61.230713576641044, 56.183690218293776],
    [-61.366266080040376, 45.09667176473487],
    [-68.40160283717302, 51.22288379742737],
    [-65.91059565227219, 46.12828610239469],
    [-69.24685336802877, 58.04159904298512],
    [-67.90746049167338, 51.458754170538626],
    [-64.69458939381633, 43.50862977964408],
    [-69.67707467644576, 55.13014562401008],
    [-67.86563100693668, 44.71454319618893],
    [-67.64200780701925, 45.346156738904824],
    [-62.784041537868234, 44.021481858972024],
    [-69.46030604589015, 50.77927933384373],
    [-67.47301069491814, 57.20749222445842],
    [-63.63826397952049, 43.06756343494853],
    [-65.03421426631598, 58.607691301001665],
    [-61.42694866295021, 44.01728814041832],
    [-65.27780516075711, 49.1577380024088],
    [-63.22809792736968, 54.75459925926561],
    [-65.75164600671681, 50.444095477144174],
    [-66.53391928446148, 46.81946313004039],
    [-61.17532370859516, 56.490308715502884],
    [-62.82596239604844, 47.62933849172895],
    [-67.91427680286654, 50.002068640008446],
    [-68.8281390917315, 41.624594548778255],
    [-62.324565557935586, 54.36070911572615],
    [-67.79789775299722, 54.032762733269294],
    [-69.90721635010229, 42.53005065790635],
    [-64.09218727436058, 44.21747980258202],
    [-65.45745182285232, 41.92888759186806],
    [-64.89743431251524, 56.02354218555687],
    [-62.74330413423213, 49.851812304102474],
    [-69.01724834169485, 43.86098277913519],
    [-65.08442833608343, 49.05799844118643],
    [-60.117926604673414, 42.56576378184112],
    [-63.380264668516965, 58.718930212178506],
    [-69.58981710789907, 56.45741811079546],
    [-69.34434256360679, 55.54353852333105],
    [-68.96303460007613, 58.368441094579225],
    [-60.499978235077876, 58.04095533357255],
    [-64.48228884995504, 52.67791964667544],
    [-63.28389119770206, 51.760879575173384],
    [-67.75617507537125, 43.75912562117327],
    [-68.00927728902863, 45.671594734286614],
    [-63.36784409964047, 44.10569333994948],
    [-65.58756228301353, 54.97295500158505],
    [-67.37277146991484, 56.476728117598626],
    [-68.10830042153275, 55.72080967470162],
    [-69.12830174563588, 45.58632852197277],
    [-64.6315835789136, 51.88413518274077],
```

(continues on next page)

(continued from previous page)

```
    [-67.75853090943711, 58.587494282518776],
    [-61.45846709902253, 53.270601672273],
    [-61.903530354456734, 57.98191726730208],
    [-65.81769750926455, 42.034979844034275],
    [-64.2364381676968, 51.762721823182545],
    [-69.75124251822547, 50.065087854580355],
    [-62.624081650070146, 49.81702630730683],
    [-64.44355690073627, 59.06794579492187],
    [-68.55204564864789, 55.336271363856184],
    [-63.25564507829077, 50.6046345112907],
    [-63.81456104951526, 40.8486627152392],
    [-64.37015352762566, 58.00983759616774],
    [-61.870172642975895, 59.15750997986467],
    [-60.02123672191712, 56.074643702400614],
    [-60.11304594655109, 43.241774996887635],
    [-66.18978478841018, 53.56622582843157],
    [-63.92923317441267, 54.78528722989218],
    [-62.48536066011403, 54.502807304022 56],
]
bbox = [-70, 40, -60, 60]
result = voronoi(points, bbox)

m = Map(center=[51.833656018902175, -64.83547210693361], zoom=4)

for point in points:
    marker = Marker(location=[point[1], point[0]])
    m.add_layer(marker)

vor = GeoJSON(name="Original", data=result, style={"color": "green"})
m.add_layer(vor)
m
```

CHAPTER

THIRTYEIGHT

LINE INTERSECT

Takes any LineString or Polygon GeoJSON and returns the intersecting point(s). If one of the Features is polygon pass the polygon feature as the first parameter to improve performance. To use this functionality Rtree or pygeos is needed to be installed.

38.1 Example

```
from geojson import LineString, Feature
from turfpy.misc import line_intersect
l1 = Feature(geometry=LineString([[126, -11], [129, -21]]))
l2 = Feature(geometry=LineString([[123, -18], [131, -14]]))
line_intersect(l1, l2)
```

```
{"features": [{"geometry": {"coordinates": [127.434783, -15.782609], "type": "Point"},
↪"properties": {}, "type": "Feature"}], "type": "FeatureCollection"}
```

38.2 Interactive Example

```
from ipyleaflet import Map, GeoJSON
from geojson import LineString, Feature
from turfpy.misc import line_intersect

m = Map(center=[-15.150511712009196, 127.52157211303712], zoom=5)

l1 = Feature(geometry=LineString([[126, -11], [129, -21]]))

l2 = Feature(geometry=LineString([[123, -18], [131, -14]]))

Line1 = GeoJSON(name="Line1", data=l1)
Line2 = GeoJSON(name="Line2", data=l2)


m.add_layer(Line1)
m.add_layer(Line2)

intersection = GeoJSON(name="intersection", data=line_intersect(l1, l2))
m.add_layer(intersection)
```

(continues on next page)

(continued from previous page)

```
m
```

CHAPTER

THIRTYNINE

LINE SEGMENT

Creates a FeatureCollection of 2-vertex LineString segments from a (Multi)LineString or (Multi)Polygon.

39.1 Example

```
from turfpy.misc import line_segment

poly = {
      "type": "Feature",
      "properties": {},
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              51.17431640625,
              47.025206001585396
            ],
            [
              45.17578125,
              43.13306116240612
            ],
            [
              54.5361328125,
              41.85319643776675
            ],
            [
              51.17431640625,
              47.025206001585396
            ]
          ]
        ]
      }
}

line_segment(poly)
```

```
{"features": [{"bbox": [45.175781, 43.133061, 51.174316, 47.025206], "geometry": {
↪"coordinates": [[45.175781, 43.133061], [51.174316, 47.025206]], "type": "LineString"},
↪ "id": 0, "properties": {}, "type": "Feature"}, {"bbox": [45.175781, 41.853196, 54.
↪536133, 43.133061], "geometry": {"coordinates": [[54.536133, 41.853196], [45.175781,
↪43.133061]], "type": "LineString"}, "id": 1, "properties": {}, "type": "Feature"}, {
↪"bbox": [51.174316, 41.853196, 54.536133, 47.025206], "geometry": {"coordinates": [[51
↪174316, 47.025206], [54.536133, 41.853196]], "type": "LineString"}, "id": 2,
↪"properties": {}, "type": "Feature"}], "type": "FeatureCollection"}
```

(continues on next page)

(continued from previous page)

39.2 Interactive Example

```
from ipyleaflet import Map, GeoJSON, LayersControl
from turfpy.misc import line_segment

m = Map(center=[44.52337579109473, 50.61581611633301], zoom=6)

poly = {
      "type": "Feature",
      "properties": {},
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              51.17431640625,
              47.025206001585396
            ],
            [
              45.17578125,
              43.13306116240612
            ],
            [
              54.5361328125,
              41.85319643776675
            ],
            [
              51.17431640625,
              47.025206001585396
            ]
          ]
        ]
      }
}

polygon = GeoJSON(name="Polygon", data=poly, style={'color':'red'})

m.add_layer(polygon)

segments = GeoJSON(name="segments", data=line_segment(poly))
m.add_layer(segments)

control = LayersControl(position='topright')
m.add_control(control)

m
```

CHAPTER

FORTY

LINE ARC

Creates a circular arc, of a circle of the given radius and center point, between bearing1 and bearing2; 0 bearing is North of center point, positive clockwise.

40.1 Example

```
from turfpy.misc import line_arc
from geojson import Feature, Point

center = Feature(geometry=Point((-75, 40)))
radius = 5
bearing1 = 25
bearing2 = 47

line_arc(center=center, radius=radius, bearing1=bearing1, bearing2=bearing2)
```

```
{"geometry": {"coordinates": [[-74.975178, 40.04075], [-74.970081, 40.03869], [-74.
↪965272, 40.036257], [-74.960799, 40.033475], [-74.957051, 40.030659]], "properties": {}
↪, "type": "LineString"}, "properties": {}, "type": "Feature"}
```

40.2 Interactive Example

```
from ipyleaflet import Map, GeoJSON, LayersControl
from turfpy.misc import line_arc
from geojson import Feature, Point, LineString, FeatureCollection

center = Feature(geometry=Point((-75, 40)))
radius = 5;
bearing1 = 25;
bearing2 = 47;

m = Map(center=[40.011313056309056, -74.97720068362348], zoom=12)

feature = line_arc(center=center, radius=radius, bearing1=bearing1, bearing2=bearing2)

fc = FeatureCollection([feature, center])
```

(continues on next page)

(continued from previous page)

```
layer = GeoJSON(name="Line_Arc", data=fc, style={'color':'red'})

m.add_layer(layer)
m
```

CHAPTER

FORTYONE

SECTOR

Creates a circular sector of a circle of given radius and center Point, between (clockwise) bearing1 and bearing2; 0 bearing is North of center point, positive clockwise.

41.1 Example

```
from turfpy.misc import sector
from geojson import Feature, Point

center = Feature(geometry=Point((-75, 40)))
radius = 5
bearing1 = 25
bearing2 = 45

feature = sector(center, radius, bearing1, bearing2, options={"properties":{"length":3}})
```

41.2 Interactive Example

```
from ipyleaflet import Map, GeoJSON, LayersControl
from turfpy.misc import sector
from geojson import Feature, Point, LineString, FeatureCollection

center = Feature(geometry=Point((-75, 40)))
radius = 5;
bearing1 = 25;
bearing2 = 45;

m = Map(center=[40.011313056309056, -74.97720068362348], zoom=12)

feature = sector(center, radius, bearing1, bearing2, options={"properties":{"length":3}})


fc = FeatureCollection([feature, center])

layer = GeoJSON(name="Line_Arc", data=fc, style={'color':'red'})

```

(continues on next page)

(continued from previous page)

```
m.add_layer(layer)
m
```

CHAPTER

FORTYTWO

RANDOM POSITION

Generates a random position, if bbox provided then the generated position will be in the bbox.

42.1 Example

```
from turfpy.random import random_position

random_position(bbox=[11.953125, 18.979025953255267, 52.03125, 46.558860303117164])
```

```
[51.329906152874294, 39.125344769112175]
```

42.2 Interactive Example

```
from geojson import Feature
from ipyleaflet import Map, GeoJSON
from turfpy.random import random_position
from turfpy.measurement import bbox_polygon

bb = [11.953125, 18.979025953255267, 52.03125, 46.558860303117164]

random_pos = random_position(bbox=bb)

f = Feature(
    geometry={
        "coordinates": random_pos,
        "type": "Point",
    }
)

geo_json = GeoJSON(name="Position", data=f)



bbox_polygon_geojson = GeoJSON(
        name="Bounding Box Polygon", data=bbox_polygon(bb), style={"color": "red"}
    )
```

(continues on next page)

(continued from previous page)

```
m = Map(center=[4.889835742990713, 5.82601547241211], zoom=1)

m.add_layer(geo_json)
m.add_layer(bbox_polygon_geojson)

m
```

CHAPTER

FORTYTHREE

RANDOM POINTS

Generates geojson random points, if bbox provided then the generated points will be in the bbox.

43.1 Example

```
from turfpy.random import random_points

random_points(count=3, bbox=[11.953125, 18.979025953255267, 52.03125, 46.
↪558860303117164])
```

```
{"features": [{"geometry": {"coordinates": [49.944089, 21.451395], "type": "Point"},
↪"properties": {}, "type": "Feature"}, {"geometry": {"coordinates": [22.941879, 21.
↪653516], "type": "Point"}, "properties": {}, "type": "Feature"}, {"geometry": {
↪"coordinates": [49.636954, 39.474522], "type": "Point"}, "properties": {}, "type":
↪"Feature"}], "type": "FeatureCollection"}
```

43.2 Interactive Example

```
from geojson import Feature
from ipyleaflet import Map, GeoJSON
from turfpy.random import random_points
from turfpy.measurement import bbox_polygon

bb = [11.953125, 18.979025953255267, 52.03125, 46.558860303117164]

fc = random_points(count=3, bbox=[11.953125, 18.979025953255267, 52.03125, 46.
↪558860303117164])

geo_json = GeoJSON(name="Points", data=fc)



bbox_polygon_geojson = GeoJSON(
        name="Bounding Box Polygon", data=bbox_polygon(bb), style={"color": "red"}
    )
```

(continues on next page)

(continued from previous page)

```
m = Map(center=[4.889835742990713, 5.82601547241211], zoom=1)

m.add_layer(geo_json)
m.add_layer(bbox_polygon_geojson)

m
```

CHAPTER

FORTYFOUR

TURFPY PACKAGE

A Python library for performing geospatial data analysis which reimplements turf.js

44.1 Subpackages

44.1.1 turfpy.dev_lib package

Submodules

turfpy.dev_lib.earcut module

This module implements earcut algorithm. This code is copied from: https://github.com/joshuaskelly/earcut-python/blob/master/earcut/earcut.py As we could not find any pip package for it.

turfpy.dev_lib.earcut.**earcut**(*data*, *hole_indices=None*, *dim=None*)

Parameters

- **data** – A list of vertices.
- **hole_indices** – A list of holes
- **dim** – An int to represent dimension.

Returns A list.

turfpy.dev_lib.earcut.**__linked_list**(*data*, *start*, *end*, *dim*, *clockwise*)

turfpy.dev_lib.earcut.**filter_points**(*start*, *end=None*)

turfpy.dev_lib.earcut.**earcut_linked**(*ear*, *triangles*, *dim*, *minx*, *miny*, *size*, *_pass=None*)

turfpy.dev_lib.earcut.**__isear**(*ear*)

turfpy.dev_lib.earcut.**__isear_hashed**(*ear*, *minx*, *miny*, *size*)

turfpy.dev_lib.earcut.**__cure_local_intersections**(*start*, *triangles*, *dim*)

turfpy.dev_lib.earcut.**__split_earcut**(*start*, *triangles*, *dim*, *minx*, *miny*, *size*)

turfpy.dev_lib.earcut.**__eliminate_holes**(*data*, *hole_indices*, *outer_node*, *dim*)

turfpy.dev_lib.earcut.**__comparex**(*a*, *b*)

turfpy.dev_lib.earcut.**__eliminatehole**(*hole*, *outer_node*)

turfpy.dev_lib.earcut.**__find_hole_bridge**(*hole*, *outer_node*)

turfpy.dev_lib.earcut.**__index_curve**(*start*, *minx*, *miny*, *size*)

turfpy.dev_lib.earcut.**__sort_linked**(*_list*)

turfpy.dev_lib.earcut.**__zorder**(*x*, *y*, *minx*, *miny*, *size*)

turfpy.dev_lib.earcut.**__get_leftmost**(*start*)

turfpy.dev_lib.earcut.**__point_in_triangle**(*ax*, *ay*, *bx*, *by*, *cx*, *cy*, *px*, *py*)

turfpy.dev_lib.earcut.**isValidDiagonal**(*a*, *b*)

turfpy.dev_lib.earcut.**area**(*p*, *q*, *r*)

turfpy.dev_lib.earcut.**equals**(*p1*, *p2*)

turfpy.dev_lib.earcut.**intersects**(*p1*, *q1*, *p2*, *q2*)

turfpy.dev_lib.earcut.**intersectsPolygon**(*a*, *b*)

turfpy.dev_lib.earcut.**locallyInside**(*a*, *b*)

turfpy.dev_lib.earcut.**middleInside**(*a*, *b*)

turfpy.dev_lib.earcut.**splitPolygon**(*a*, *b*)

turfpy.dev_lib.earcut.**insertNode**(*i*, *x*, *y*, *last*)

turfpy.dev_lib.earcut.**__remove_node**(*p*)

class turfpy.dev_lib.earcut.**Node**(*i*, *x*, *y*)

Bases: object

__init__(*i*, *x*, *y*)

turfpy.dev_lib.earcut.**__deviation**(*data*, *hole_indices*, *dim*, *triangles*)

turfpy.dev_lib.earcut.**__signed_area**(*data*, *start*, *end*, *dim*)

turfpy.dev_lib.earcut.**__flatten**(*data*)

turfpy.dev_lib.earcut.**__unflatten**(*data*)

turfpy.dev_lib.spline module

class turfpy.dev_lib.spline.**Spline**(*points_data=[]*, *resolution=10000*, *sharpness=0.85*)

Bases: object

__init__(*points_data=[]*, *resolution=10000*, *sharpness=0.85*)

cache_steps(*mindist*)

pos(*time*)

bezier(*t*, *p1*, *c1*, *c2*, *p2*)

B(*t*)

44.2 Submodules

44.2.1 turfpy.extra module

44.2.2 turfpy.helper module

turfpy.helper.**convert_length**(*length*, *original_unit='km'*, *final_unit='km'*)
#TODO: Add description

Parameters

- **original_unit** (*str*) –
- **final_unit** (*str*) –

turfpy.helper.**length_to_radians**(*distance*, *unit='km'*)
#TODO: Add description

Parameters **unit** (*str*) –

turfpy.helper.**radians_to_length**(*radians*, *unit='km'*)
#TODO: Add description

Parameters **unit** (*str*) –

turfpy.helper.**get_type**(*geojson*)
#TODO: Add description

turfpy.helper.**get_coord**(*coord*)
#TODO: Add description

turfpy.helper.**get_geom**(*geojson*)
#TODO: Add description

turfpy.helper.**get_coords**(*coords*)
#TODO: Add description

turfpy.helper.**feature_of**(*feature*, *ttype*, *name*)
#TODO: Add description

turfpy.helper.**length_to_degrees**(*distance*, *units='km'*)
#TODO: Add description

Parameters **units** (*str*) –

turfpy.helper.**radians_to_degrees**(*radians*)
#TODO: Add description

Parameters **radians** (*float*) –

turfpy.helper.**convert_angle_to_360**(*alfa*)

Parameters **alfa** (*float*) –

44.2.3 turfpy.measurement module

This module implements some of the spatial analysis techniques and processes used to understand the patterns and relationships of geographic features. This is mainly inspired by turf.js. link: http://turfjs.org/

turfpy.measurement.**bearing**(*start*, *end*, *final=False*)

Takes two Point and finds the geographic bearing between them.

Parameters

- **start** (*geojson.feature.Feature*) – A object of `Point` to represent start point.
- **end** (*geojson.feature.Feature*) – A object of `Point` to represent end point.
- **final** – A boolean calculates the final bearing if True.

Returns A float calculated bearing.

Return type float

Example:

```
>>> from geojson import Point, Feature
>>> from turfpy import measurement
>>> start = Feature(geometry=Point((-75.343, 39.984)))
>>> end = Feature(geometry=Point((-75.534, 39.123)))
>>> measurement.bearing(start,end)
```

turfpy.measurement.**_calculate_final_bearing**(*start*, *end*)

#TODO: Add description

Return type float

turfpy.measurement.**distance**(*point1*, *point2*, *units='km'*)

Calculates distance between two Points. A point is containing latitude and logitude in decimal degrees and `unit` is optional.

It calculates distance in units such as kilometers, meters, miles, feet and inches.

Parameters

- **point1** (*geojson.feature.Feature*) – first point; tuple of (latitude, longitude) in decimal degrees.
- **point2** (*geojson.feature.Feature*) – second point; tuple of (latitude, longitude) in decimal degrees.
- **units** (*str*) – A string containing unit, E.g. kilometers = 'km', miles = 'mi', meters = 'm', feet = 'ft', inches = 'in'.

Returns The distance between the two points in the requested unit, as a float.

Example:

```
>>> from turfpy import measurement
>>> from geojson import Point, Feature
>>> start = Feature(geometry=Point((-75.343, 39.984)))
>>> end = Feature(geometry=Point((-75.534, 39.123)))
>>> measurement.distance(start,end)
```

turfpy.measurement.**area**(*geojson*)

This function calculates the area of the Geojson object given as input.

Parameters **geojson** (*Union[geojson.geometry.Point, geojson.geometry.LineString, geojson.geometry.Polygon, geojson.geometry.MultiPoint, geojson.geometry.MultiLineString, geojson.geometry.MultiPolygon, geojson.feature.Feature, geojson.feature.FeatureCollection]*) – Geojson object for which area is to be found.

Returns area for the given Geojson object in square meters.

Example:

```
>>> from turfpy.measurement import area
>>> from geojson import Feature, FeatureCollection
>>> geometry_1 = {"coordinates": [[[0, 0], [0, 10], [10, 10], [10, 0], [0, 0]]],
↪"type": "Polygon"}  # noqa E501
>>> geometry_2 = {"coordinates": [[[2.38, 57.322], [23.194, -20.28], [-120.43, 19.
↪15],[2.38, 57.322]]], "type": "Polygon"}  # noqa E501
>>> feature_1 = Feature(geometry=geometry_1)
>>> feature_2 = Feature(geometry=geometry_2)
>>> feature_collection = FeatureCollection([feature_1, feature_2])
```

```
>>> area(feature_collection)
```

turfpy.measurement.**bbox**(*geojson*)

This function is used to generate bounding box coordinates for given geojson.

Parameters **geojson** – Geojson object for which bounding box is to be found.

Returns bounding box for the given Geojson object.

Example :

```
>>> from turfpy.measurement import bbox
>>> from geojson import Polygon
```

```
>>> p = Polygon([(2.38, 57.322), (23.194, -20.28), (-120.43, 19.15),(2.38, 57.322)])
>>> bb = bbox(p)
```

turfpy.measurement.**bbox_polygon**(*bbox*, *properties={}*)

To generate a Polygon Feature for the bounding box generated using bbox.

Parameters

- **bbox** (*list*) – bounding box generated for a geojson.
- **properties** (*dict*) – properties to be added to the returned feature.

Returns polygon for the given bounding box coordinates.

Return type geojson.feature.Feature

Example :

```
>>> from turfpy.measurement import bbox_polygon, bbox
>>> from geojson import Polygon
```

```
>>> p = Polygon([((2.38, 57.322), (23.194, -20.28), (-120.43, 19.15),
... (2.38, 57.322))])
>>> bb = bbox(p)
>>> feature = bbox_polygon(bb)
```

turfpy.measurement.**center**(*geojson*, *properties=None*)

Takes a Feature or FeatureCollection and returns the absolute center point of all features.

Parameters

- **geojson** – GeoJSON for which centered to be calculated.
- **properties** (*Optional[dict]*) – Optional parameters to be set to the generated feature.

Returns Point feature for the center.

Return type geojson.feature.Feature

Example :

```
>>> from turfpy.measurement import center
>>> from geojson import Feature, FeatureCollection, Point
```

```
>>> f1 = Feature(geometry=Point((-97.522259, 35.4691)))
>>> f2 = Feature(geometry=Point((-97.502754, 35.463455)))
>>> f3 = Feature(geometry=Point((-97.508269, 35.463245)))
>>> feature_collection = FeatureCollection([f1, f2, f3])
>>> feature = center(feature_collection)
```

turfpy.measurement.**envelope**(*geojson*)

Takes any number of features and returns a rectangular Polygon that encompasses all vertices.

Parameters **geojson** – geojson input features for which envelope to be generated.

Returns returns envelope i.e bounding box polygon.

Return type geojson.feature.Feature

Example :

```
>>> from turfpy.measurement import envelope
>>> from geojson import Feature, FeatureCollection, Point
```

```
>>> f1 = Feature(geometry=Point((-97.522259, 35.4691)))
>>> f2 = Feature(geometry=Point((-97.502754, 35.463455)))
>>> f3 = Feature(geometry=Point((-97.508269, 35.463245)))
>>> feature_collection = FeatureCollection([f1, f2, f3])
>>> feature = envelope(feature_collection)
```

turfpy.measurement.**length**(*geojson*, *units='km'*)

Takes a geojson and measures its length in the specified units.

Parameters

- **geojson** – geojson for which the length is to be determined.
- **units** (*str*) – units in which length is to be returned.

Returns length of the geojson in specified units.

Example:

```
>>> from turfpy.measurement import length
>>> from geojson import LineString
>>> ls = LineString([(115, -32), (131, -22), (143, -25), (150, -34)])
>>> length(ls)
```

turfpy.measurement.**destination**(*origin*, *distance*, *bearing*, *options={}*)

Takes a Point and calculates the location of a destination point given a distance in degrees, radians, miles, or kilometers and bearing in degrees.

Parameters

- **origin** (*Union[geojson.feature.Feature, geojson.geometry.Point]*) – Start point.
- **distance** – distance upto which the destination is from origin.
- **bearing** – Direction in which is the destination is from origin.
- **options** (*dict*) – Option like units of distance and properties to be passed to destination point feature, value for units are 'mi', 'km', 'deg' and 'rad'.

Returns Feature: destination point in at the given distance and given direction.

Return type geojson.feature.Feature

Example:

```
>>> from turfpy.measurement import destination
>>> from geojson import Point, Feature
>>> origin = Feature(geometry=Point([-75.343, 39.984]))
>>> distance = 50
>>> bearing = 90
>>> options = {'units': 'mi'}
>>> destination(origin,distance,bearing,options)
```

turfpy.measurement.**centroid**(*geojson*, *properties=None*)

Takes one or more features and calculates the centroid using the mean of all vertices.

Parameters

- **geojson** – Input features
- **properties** (*Optional[dict]*) – Properties to be set to the output Feature point

Returns Feature: Point feature which is the centroid of the given features

Return type geojson.feature.Feature

Example:

```
>>> from turfpy.measurement import centroid
>>> from geojson import Polygon
>>> polygon = Polygon([((-81, 41), (-88, 36), (-84, 31), (-80, 33), (-77, 39),
(-81, 41))])
>>> centroid(polygon)
```

turfpy.measurement.**along**(*line*, *dist*, *unit='km'*)

This function is used identify a Point at a specified distance along a LineString.

Parameters

- **line** (*geojson.feature.Feature*) – LineString on which the point to be identified
- **dist** – Distance from the start of the LineString
- **unit** (*str*) – unit of distance

Returns Feature : Point at the distance on the LineString passed

Return type geojson.feature.Feature

Example :

```
>>> from turfpy.measurement import along
>>> from geojson import LineString, Feature
>>> ls = Feature(geometry=LineString([(-83, 30), (-84, 36), (-78, 41)]))
>>> along(ls,200,'mi')
```

turfpy.measurement.**midpoint**(*point1*, *point2*)

This function is used to get midpoint between any the two points.

Parameters

- **point1** (*geojson.feature.Feature*) – First point.
- **point2** (*geojson.feature.Feature*) – Second point.

Returns Feature: Point which is the midpoint of the two points given as input.

Return type geojson.feature.Feature

Example:

```
>>> from turfpy.measurement import midpoint
>>> from geojson import Point, Feature
>>> point1 = Feature(geometry=Point((144.834823, -37.771257)))
>>> point2 = Feature(geometry=Point((145.14244, -37.830937)))
>>> midpoint(point1, point2)
```

turfpy.measurement.**nearest_point**(*target_point*, *points*)

Takes a reference Point Feature and FeatureCollection of point features and returns the point from the FeatureCollection closest to the reference Point Feature.

Parameters

- **target_point** (*geojson.feature.Feature*) – Feature Point of reference.
- **points** (*geojson.feature.FeatureCollection*) – FeatureCollection of points.

Returns a Point Feature from the FeatureCollection which is closest to the reference Point.

Return type geojson.feature.Feature

Example:

```
>>> from turfpy.measurement import nearest_point
>>> from geojson import Point, Feature, FeatureCollection
>>> f1 = Feature(geometry=Point((28.96991729736328,41.01190001748873)))
>>> f2 = Feature(geometry=Point((28.948459, 41.024204)))
>>> f3 = Feature(geometry=Point((28.938674, 41.013324)))
>>> fc = FeatureCollection([f1, f2 ,f3])
>>> t = Feature(geometry=Point((28.973865, 41.011122)))
>>> nearest_point(t ,fc)
```

turfpy.measurement.**point_on_feature**(*geojson*)

Takes a Feature or FeatureCollection and returns a Point guaranteed to be on the surface of the feature.

Parameters **geojson** – Feature or FeatureCollection on which the Point is to be found.

Returns Feature point which on the provided feature.

Return type geojson.feature.Feature

Example:

```
>>> from turfpy.measurement import point_on_feature
>>> from geojson import  Polygon, Feature
>>> point = Polygon([((116, -36), (131, -32), (146, -43), (155, -25), (133, -9),
(111, -22), (116, -36))])
>>> feature = Feature(geometry=point)
>>> point_on_feature(feature)
```

turfpy.measurement.**_normalize**(*geojson*)

turfpy.measurement.**_point_on_segment**(*x*, *y*, *x1*, *y1*, *x2*, *y2*)

turfpy.measurement.**boolean_point_in_polygon**(*point*, *polygon*, *ignore_boundary=False*)

Takes a Point or a Point Feature and Polygon or Polygon Feature as input and returns True if Point is in given Feature.

Parameters

- **point** – Point or Point Feature.
- **polygon** – Polygon or Polygon Feature.
- **ignore_boundary** – [Optional] default value is False, specify whether to exclude boundary of the given polygon or not.

Returns True if the given Point is in Polygons else False

Example:

```
>>> from turfpy.measurement import boolean_point_in_polygon
>>> from geojson import Point, MultiPolygon, Feature
>>> point = Feature(geometry=Point((-77, 44)))
>>> polygon = Feature(geometry=MultiPolygon([([(-81, 41), (-81, 47), (-72, 47),
(-72, 41), (-81, 41)],),
>>> ([(3.78, 9.28), (-130.91, 1.52), (35.12, 72.234), (3.78, 9.28)],)]))
>>> boolean_point_in_polygon(point, polygon)
```

turfpy.measurement.**in_ring**(*pt*, *ring*, *ignore_boundary*)

turfpy.measurement.**in_bbox**(*pt*, *bbox*)

turfpy.measurement.**explode**(*geojson*)

turfpy.measurement.**polygon_tangents**(*point*, *polygon*)

Finds the tangents of a (Multi)Polygon from a Point.

Parameters

- **point** – Point or Point Feature.
- **polygon** – (Multi)Polygon or (Multi)Polygon Feature.

Returns FeatureCollection of two tangent Point Feature.

Example:

```
>>> from turfpy.measurement import polygon_tangents
>>> from geojson import Polygon, Point, Feature
>>> point = Feature(geometry=Point((61, 5)))
>>> polygon = Feature(geometry=Polygon([(11, 0), (22, 4), (31, 0), (31, 11),
```

(continues on next page)

(continued from previous page)

```
...                                 (21, 15), (11, 11), (11, 0)]))
>>> polygon_tangents(point, polygon)
```

turfpy.measurement.**process_polygon**(*polygon_coords*, *pt_coords*, *eprev*, *enext*, *rtan*, *ltan*)

turfpy.measurement.**_is_above**(*point1*, *point2*, *point3*)

turfpy.measurement.**_is_below**(*point1*, *point2*, *point3*)

turfpy.measurement.**_is_left**(*point1*, *point2*, *point3*)

turfpy.measurement.**point_to_line_distance**(*point*, *line*, *units='km'*, *method='geodesic'*)

Returns the minimum distance between a Point and any segment of the LineString.

Parameters

- **point** (*geojson.feature.Feature*) – Point Feature from which distance to be measured.
- **line** (*geojson.feature.Feature*) – Point LineString from which distance to be measured.
- **units** – units for distance 'km', 'm', 'mi, 'ft', 'in', 'deg', 'cen', 'rad', 'naut', 'yd'
- **method** – Method which is used to calculate, values can be 'geodesic' or 'planar'

Returns Approximate distance between the LineString and Point

Example:

```
>>> from turfpy.measurement import point_to_line_distance
>>> from geojson import LineString, Point, Feature
>>> point = Feature(geometry=Point((0, 0)))
>>> linestring = Feature(geometry=LineString([(1, 1),(-1, 1)]))
>>> point_to_line_distance(point, linestring)
```

turfpy.measurement.**distance_to_segment**(*p*, *a*, *b*, *options*)

turfpy.measurement.**_calc_distance**(*a*, *b*, *options*)

turfpy.measurement.**_dot**(*u*, *v*)

turfpy.measurement.**rhumb_bearing**(*start*, *end*, *final=False*)

Takes two points and finds the bearing angle between them along a Rhumb line, i.e. the angle measured in degrees start the north line (0 degrees).

Parameters

- **start** – Start Point or Point Feature.
- **end** – End Point or Point Feature.
- **final** – Calculates the final bearing if true

Returns bearing from north in decimal degrees, between -180 and 180 degrees (positive clockwise)

Example:

```
>>> from turfpy.measurement import rhumb_bearing
>>> from geojson import Feature, Point
>>> start = Feature(geometry=Point((-75.343, 39.984)))
>>> end = Feature(geometry=Point((-75.534, 39.123)))
>>> rhumb_bearing(start, end, True)
```

turfpy.measurement.**calculate_rhumb_bearing**(*fro*, *to*)
#TODO: Add description

turfpy.measurement.**rhumb_destination**(*origin*, *distance*, *bearing*, *options={}*)
Returns the destination Point having travelled the given distance along a Rhumb line from the origin Point with the (varant) given bearing.

Parameters

- **origin** – Starting Point
- **distance** – Distance from the starting point
- **bearing** – Varant bearing angle ranging from -180 to 180 degrees from north
- **options** (*dict*) – A dict of two values 'units' for the units of distance provided and 'properties' that are to be passed to the Destination Feature Point Example :- {'units':'mi', 'properties': {"marker-color": "F00"}}

Returns Destination Feature Point

Return type geojson.feature.Feature

Example:

```
>>> from turfpy.measurement import rhumb_destination
>>> from geojson import Point, Feature
>>> start = Feature(geometry=Point((-75.343, 39.984)),
properties={"marker-color": "F00"})
>>> distance = 50
>>> bearing = 90
>>> rhumb_destination(start, distance, bearing, {'units':'mi',
'properties': {"marker-color": "F00"}})
```

turfpy.measurement.**_calculate_rhumb_destination**(*origin*, *distance*, *bearing*, *radius=None*)

turfpy.measurement.**rhumb_distance**(*start*, *to*, *units='km'*)
Calculates the distance along a rhumb line between two points in degrees, radians, miles, or kilometers.

Parameters

- **start** – Start Point or Point Feature from which distance to be calculated.
- **to** – End Point or Point Feature upto which distance to be calculated.
- **units** – Units in which distance to be calculated, values can be 'deg', 'rad', 'mi', 'km'

Returns Distance calculated from provided start to end Point.

Example:

```
>>> from turfpy.measurement import rhumb_distance
>>> from geojson import Point, Feature
>>> start = Feature(geometry=Point((-75.343, 39.984)))
>>> end = Feature(geometry=Point((-75.534, 39.123)))
>>> rhumb_distance(start, end,'mi')
```

turfpy.measurement.**_calculate_rhumb_distance**(*origin*, *destination_point*, *radius=None*)

turfpy.measurement.**square**(*bbox*)
Takes a bounding box and calculates the minimum square bounding box that would contain the input.

Parameters **bbox** (*list*) – Bounding box extent in west, south, east, north order

Returns A square surrounding bbox

Example:

```
>>> from turfpy.measurement import square
>>> bbox = [-20, -20, -15, 0]
>>> square(bbox)
```

turfpy.measurement.**points_within_polygon**(*points*, *polygons*, *chunk_size=1*)

Find Point(s) that fall within (Multi)Polygon(s).

This function takes two inputs GeoJSON Feature `geojson.Point` or `geojson.FeatureCollection` of Points and GeoJSON Feature `geojson.Polygon` or Feature `geojson.MultiPolygon` or FeatureCollection of `geojson.Polygon` or Feature `geojson.MultiPolygon`. and returns all points with in in those Polygon(s) or (Multi)Polygon(s).

Parameters

- **points** (*Union[geojson.feature.Feature, geojson.feature.FeatureCollection]*) – A single GeoJSON `Point` feature or FeatureCollection of Points.
- **polygons** (*Union[geojson.feature.Feature, geojson.feature.FeatureCollection]*) – A Single GeoJSON Polygon/MultiPolygon or FeatureCollection of Polygons/MultiPolygons.
- **chunk_size** (*int*) – Number of chunks each process to handle. The default value is 1, for a large number of features please use *chunk_size* greater than 1 to get better results in terms of performance.

Returns A `geojson.FeatureCollection` of Points.

Return type geojson.feature.FeatureCollection

turfpy.measurement.**check_each_point**(*point*, *polygons*, *results*)

44.2.4 turfpy.meta module

turfpy.meta.**geom_reduce**(*geojson*, *initial_value_param*)

turfpy.meta.**geom_each**(*geojson*, *callback*)

Iterate over each geometry in any GeoJSON object, similar to Array.forEach().

Parameters

- **geojson** – Point|Polygon|MultiPolygon|MultiPoint|LineString|MultiLineString| FeatureCollection|Feature geojson any GeoJSON object
- **callback** –

Returns

turfpy.meta.**calculate_area**(*geom*)

Return type float

turfpy.meta.**polygon_area**(*coords*)

Parameters **coords** (*list*) –

Return type float

turfpy.meta.**ring_area**(*coords*)

Parameters **coords** (*list*) –

turfpy.meta.**rad**(*num*)

Parameters **num** (*float*) –

turfpy.meta.**coord_each**(*geojson*, *callback*, *excludeWrapCoord=None*)
Iterate over coordinates in any GeoJSON object, similar to Array.forEach() :return:

turfpy.meta.**segment_reduce**(*geojson*, *callback*, *initial_value=None*)

turfpy.meta.**segment_each**(*geojson*, *callback*)

turfpy.meta.**flatten_each**(*geojson*, *callback*)

turfpy.meta.**feature_each**(*geojson*, *callback*)

44.2.5 turfpy.transformation module

This module implements some of the spatial analysis techniques and processes used to understand the patterns and relationships of geographic features. This is mainly inspired by turf.js. link: http://turfjs.org/

turfpy.transformation.**circle**(*center*, *radius*, *steps=64*, *units='km'*, ***kwargs*)
Takes a Point and calculates the circle polygon given a radius in degrees, radians, miles, or kilometers; and steps for precision.

Parameters

- **center** (*geojson.feature.Feature*) – A *Point* object representing center point of circle.
- **radius** (*int*) – An int representing radius of the circle.
- **steps** (*int*) – An int representing number of steps.
- **units** (*str*) – A string representing units of distance e.g. 'mi', 'km', 'deg' and 'rad'.
- **kwargs** – A dict representing additional properties.

Returns A polygon feature object.

Return type geojson.geometry.Polygon

Example:

```
>>> from turfpy.transformation import circle
>>> from geojson import Feature, Point
>>> circle(center=Feature(geometry=Point((-75.343, 39.984))), radius=5, steps=10)
```

turfpy.transformation.**bbox_clip**(*geojson*, *bbox*)
Takes a Feature or geometry and a bbox and clips the feature to the bbox :param geojson: Geojson data :param bbox: Bounding Box which is used to clip the geojson :return: Clipped geojson

Example:

```
>>> from turfpy.transformation import bbox_clip
>>> from geojson import Feature
>>> f = Feature(geometry={"coordinates": [[[2, 2], [8, 4],
>>> [12, 8], [3, 7], [2, 2]]], "type": "Polygon"})
```

(continues on next page)

(continued from previous page)

```
>>> bbox = [0, 0, 10, 10]
>>> clip = bbox_clip(f, bbox)
```

Parameters

- **geojson** (*geojson.feature.Feature*) –
- **bbox** (*list*) –

Return type geojson.feature.Feature

turfpy.transformation.**intersect**(*features*)

Takes polygons and finds their intersection :param features: List of features of Feature Collection :return: Intersection Geojson Feature

Example:

```
>>> from turfpy.transformation import intersect
>>> from geojson import Feature
>>> f = Feature(geometry={"coordinates": [
>>> [[-122.801742, 45.48565], [-122.801742, 45.60491],
>>> [-122.584762, 45.60491], [-122.584762, 45.48565],
>>> [-122.801742, 45.48565]]], "type": "Polygon"})
>>> b = Feature(geometry={"coordinates": [
>>> [[-122.520217, 45.535693], [-122.64038, 45.553967],
>>> [-122.720031, 45.526554], [-122.669906, 45.507309],
>>> [-122.723464, 45.446643], [-122.532577, 45.408574],
>>> [-122.487258, 45.477466], [-122.520217, 45.535693]
>>> ]], "type": "Polygon"})
>>> inter = intersect([f, b])
```

Parameters **features** (*Union[List[geojson.feature.Feature], geojson.feature.FeatureCollection]*) –

Return type geojson.feature.Feature

turfpy.transformation.**bezier_spline**(*line*, *resolution=10000*, *sharpness=0.85*)

Takes a line and returns a curved version by applying a Bezier spline algorithm :param line: LineString Feature which is used to draw the curve :param resolution: time in milliseconds between points :param sharpness: a measure of how curvy the path should be between splines :return: Curve as LineString Feature

Example:

```
>>> from geojson import LineString, Feature
>>> from turfpy.transformation import bezier_spline
>>> ls = LineString([(-76.091308, 18.427501),
>>>                  (-76.695556, 18.729501),
>>>                  (-76.552734, 19.40443),
>>>                  (-74.61914, 19.134789),
>>>                  (-73.652343, 20.07657),
>>>                  (-73.157958, 20.210656)])
>>> f = Feature(geometry=ls)
>>> bezier_spline(f)
```

Parameters **line** (*geojson.feature.Feature*) –

Return type geojson.feature.Feature

turfpy.transformation.**merge_dict**(*dicts*)

Parameters **dicts** (*list*) –

turfpy.transformation.**union**(*features*)

Given list of features or `FeatureCollection` return union of those.

Parameters **features** (*Union[List[geojson.feature.Feature], geojson.feature.FeatureCollection]*) – A list of GeoJSON features or FeatureCollection.

Returns A GeoJSON Feature or FeatureCollection.

Return type Union[geojson.feature.Feature, geojson.feature.FeatureCollection]

Example:

```
>>> from turfpy.transformation import union
>>> from geojson import Feature, Polygon, FeatureCollection
>>> f1 = Feature(geometry=Polygon([[
...           [-82.574787, 35.594087],
...           [-82.574787, 35.615581],
...           [-82.545261, 35.615581],
...           [-82.545261, 35.594087],
...           [-82.574787, 35.594087]
...       ]]), properties={"fill": "#00f"})
>>> f2 = Feature(geometry=Polygon([[
...           [-82.560024, 35.585153],
...           [-82.560024, 35.602602],
...           [-82.52964, 35.602602],
...           [-82.52964, 35.585153],
...           [-82.560024, 35.585153]]]), properties={"fill": "#00f"})
>>> union(FeatureCollection([f1, f2], properties={"combine": "yes"}))
```

turfpy.transformation.**_alpha_shape**(*points*, *alpha*)

Compute the alpha shape (concave hull) of a set of points.

Parameters

- **points** – Iterable container of points.
- **alpha** – alpha value to influence the gooeyness of the border. Smaller numbers don't fall inward as much as larger numbers. Too large, and you lose everything!

turfpy.transformation.**get_points**(*features*)

turfpy.transformation.**get_ext_points**(*geom*, *points*)

turfpy.transformation.**concave**(*features*, *alpha=2*)

Generate concave hull for the given feature or Feature Collection.

Parameters

- **features** (*Union[geojson.feature.Feature, geojson.feature.FeatureCollection]*) – It can be a feature or Feature Collection
- **alpha** – Alpha determines the shape of concave hull, greater values will make shape more tighten

Returns Feature of concave hull polygon

Example:

```
>>> from turfpy.transformation import concave
>>> from geojson import FeatureCollection, Feature, Point
>>> f1 = Feature(geometry=Point((-63.601226, 44.642643)))
>>> f2 = Feature(geometry=Point((-63.591442, 44.651436)))
>>> f3 = Feature(geometry=Point((-63.580799, 44.648749)))
>>> f4 = Feature(geometry=Point((-63.573589, 44.641788)))
>>> f5 = Feature(geometry=Point((-63.587665, 44.64533)))
>>> f6 = Feature(geometry=Point((-63.595218, 44.64765)))
>>> fc = [f1, f2, f3, f4, f5, f6]
>>> concave(FeatureCollection(fc), alpha=100)
```

turfpy.transformation.**convex**(*features*)

Generate convex hull for the given feature or Feature Collection

Parameters **features** (*Union[geojson.feature.Feature, geojson.feature.FeatureCollection]*) – It can be a feature or Feature Collection

Returns Feature of convex hull polygon

Example:

```
>>> from turfpy.transformation import convex
>>> from geojson import FeatureCollection, Feature, Point
>>> f1 = Feature(geometry=Point((10.195312, 43.755225)))
>>> f2 = Feature(geometry=Point((10.404052, 43.8424511)))
>>> f3 = Feature(geometry=Point((10.579833, 43.659924)))
>>> f4 = Feature(geometry=Point((10.360107, 43.516688)))
>>> f5 = Feature(geometry=Point((10.14038, 43.588348)))
>>> f6 = Feature(geometry=Point((10.195312, 43.755225)))
>>> fc = [f1, f2, f3, f4, f5, f6]
>>> convex(FeatureCollection(fc))
```

turfpy.transformation.**dissolve**(*features*, *property_name=None*)

Take FeatureCollection or list of features to dissolve based on property_name provided. :param features: A list of GeoJSON features or FeatureCollection. :param property_name: Name of property based on which to dissolve. :return: A GeoJSON Feature or FeatureCollection.

Example:

```
>>> from geojson import Polygon, Feature, FeatureCollection
>>> from turfpy.transformation import dissolve
>>> f1 = Feature(geometry=Polygon([[
>>>     [0, 0],
>>>     [0, 1],
>>>     [1, 1],
>>>     [1, 0],
>>>     [0, 0]]]), properties={"combine": "yes", "fill": "#00f"})
>>> f2 = Feature(geometry=Polygon([[
>>>     [0, -1],
>>>     [0, 0],
>>>     [1, 0],
>>>     [1, -1],
```

(continues on next page)

(continued from previous page)

```
>>>     [0,-1]]]), properties={"combine": "yes"})
>>> f3 = Feature(geometry=Polygon([[
>>>     [1,-1],
>>>     [1, 0],
>>>     [2, 0],
>>>     [2, -1],
>>>     [1, -1]]]), properties={"combine": "no"})
>>> dissolve(FeatureCollection([f1, f2, f3]), property_name='combine')
```

Parameters

- **features** (*Union[List[geojson.feature.Feature], geojson.feature.FeatureCollection]*) –
- **property_name** (*Optional[str]*) –

Return type geojson.feature.FeatureCollection

turfpy.transformation.**difference**(*feature_1*, *feature_2*)

Find the difference between given two features. :param feature_1: A GeoJSON feature :param feature_2: A GeoJSON feature :return: A GeoJSON feature

Example:

```
>>> from geojson import Polygon, Feature
>>> from turfpy.transformation import difference
>>> f1 = Feature(geometry=Polygon([[
>>>     [128, -26],
>>>     [141, -26],
>>>     [141, -21],
>>>     [128, -21],
>>>     [128, -26]]]), properties={"combine": "yes", "fill": "#00f"})
>>> f2 = Feature(geometry=Polygon([[
>>>     [126, -28],
>>>     [140, -28],
>>>     [140, -20],
>>>     [126, -20],
>>>     [126, -28]]]), properties={"combine": "yes"})
>>> difference(f1, f2)
```

Parameters

- **feature_1** (*geojson.feature.Feature*) –
- **feature_2** (*geojson.feature.Feature*) –

Return type geojson.feature.Feature

turfpy.transformation.**transform_rotate**(*feature*, *angle*, *pivot=None*, *mutate=False*)

Rotates any geojson Feature or Geometry of a specified angle, around its centroid or a given pivot point; all rotations follow the right-hand rule.

Parameters

- **feature** (*Union[List[geojson.feature.Feature], geojson.feature.FeatureCollection]*) – Geojson to be rotated.

- **angle** (*float*) – angle of rotation (along the vertical axis), from North in decimal degrees, negative clockwise
- **pivot** (*Optional[list]*) – point around which the rotation will be performed
- **mutate** (*bool*) – allows GeoJSON input to be mutated (significant performance increase if True)

Returns the rotated GeoJSON

Example :-

```
>>> from turfpy.transformation import transform_rotate
>>> from geojson import Polygon, Feature
>>> f = Feature(geometry=Polygon([[[0,29],[3.5,29],[2.5,32],[0,29]]]))
>>> pivot = [0, 25]
>>> transform_rotate(f, 10, pivot)
```

turfpy.transformation.**transform_translate**(*feature*, *distance*, *direction*, *units='km'*, *z_translation=0*, *mutate=False*)

Moves any geojson Feature or Geometry of a specified distance along a Rhumb Line on the provided direction angle.

Parameters

- **feature** (*Union[List[geojson.feature.Feature], geojson.feature.FeatureCollection]*) – Geojson data that is to be translated
- **distance** (*float*) – length of the motion; negative values determine motion in opposite direction
- **direction** (*float*) – of the motion; angle from North in decimal degrees, positive clockwise
- **units** (*str*) – units for the distance and z_translation
- **z_translation** (*float*) – length of the vertical motion, same unit of distance
- **mutate** (*bool*) – allows GeoJSON input to be mutated (significant performance increase if true)

Returns the translated GeoJSON

Example :-

```
>>> from turfpy.transformation import transform_translate
>>> from geojson import Polygon, Feature
>>> f = Feature(geometry=Polygon([[[0,29],[3.5,29],[2.5,32],[0,29]]]))
>>> transform_translate(f, 100, 35, mutate=True)
```

turfpy.transformation.**transform_scale**(*features*, *factor*, *origin='centroid'*, *mutate=False*)

Scale a GeoJSON from a given point by a factor of scaling (ex: factor=2 would make the GeoJSON 200% larger). If a FeatureCollection is provided, the origin point will be calculated based on each individual Feature.

Parameters

- **features** – GeoJSON to be scaled
- **factor** (*float*) – of scaling, positive or negative values greater than 0
- **origin** (*Union[str, list]*) – Point from which the scaling will occur (string options: sw/se/nw/ne/center/centroid)

- **mutate** (*bool*) – allows GeoJSON input to be mutated (significant performance increase if true)

Returns Scaled Geojson

Example :-

```
>>> from turfpy.transformation import transform_scale
>>> from geojson import Polygon, Feature
>>> f = Feature(geometry=Polygon([[[0,29],[3.5,29],[2.5,32],[0,29]]]))
>>> transform_scale(f, 3, origin=[0, 29])
```

turfpy.transformation.**scale**(*feature*, *factor*, *origin*)

turfpy.transformation.**define_origin**(*geojson*, *origin*)

turfpy.transformation.**tesselate**(*poly*)

Tesselates a Feature into a FeatureCollection of triangles using earcut.

Parameters **poly** (*geojson.feature.Feature*) – A GeoJSON feature class:geojson.Polygon.

Returns A GeoJSON FeatureCollection of triangular polygons.

Return type geojson.feature.FeatureCollection

Example:

```
>>> from geojson import Feature
>>> from turfpy.transformation import tesselate
>>> polygon = Feature(geometry={"coordinates": [[[11, 0], [22, 4], [31, 0], [31,␣
↪11],
... [21, 15], [11, 11], [11, 0]]], "type": "Polygon"})
>>> tesselate(polygon)
```

turfpy.transformation.**__process_polygon**(*coordinates*)

turfpy.transformation.**__flatten_coords**(*data*)

turfpy.transformation.**line_offset**(*geojson*, *distance*, *unit='km'*)

Takes a linestring or multilinestring and returns a line at offset by the specified distance.

Parameters

- **geojson** (*geojson.feature.Feature*) – input GeoJSON
- **distance** (*float*) – distance to offset the line (can be of negative value)
- **unit** (*str*) – Units in which distance to be calculated, values can be 'deg', 'rad', 'mi', 'km', default is 'km'

Returns Line feature offset from the input line

Return type geojson.feature.Feature

Example:

```
>>> from geojson import MultiLineString, Feature
>>> from turfpy.transformation import line_offset
>>> ls = Feature(geometry=MultiLineString([
... [(3.75, 9.25), (-130.95, 1.52)],
... [(23.15, -34.25), (-1.35, -4.65), (3.45, 77.95)]
```

(continues on next page)

(continued from previous page)

```
... ]))
>>> line_offset(ls, 2, unit='mi')
```

turfpy.transformation.**line_offset_feature**(*line*, *distance*, *units*)

turfpy.transformation.**_process_segment**(*point1*, *point2*, *offset*)

turfpy.transformation.**_intersection**(*a*, *b*)

turfpy.transformation.**_is_parallel**(*a*, *b*)

turfpy.transformation.**_ab**(*segment*)

turfpy.transformation.**_cross_product**(*v1*, *v2*)

turfpy.transformation.**_intersect_segments**(*a*, *b*)

turfpy.transformation.**_add**(*v1*, *v2*)

turfpy.transformation.**_sub**(*v1*, *v2*)

turfpy.transformation.**_scalar_mult**(*s*, *v*)

turfpy.transformation.**voronoi**(*points*, *bbox=None*)

Takes a FeatureCollection of points, and a bounding box, and returns a FeatureCollection of Voronoi polygons.

Parameters

- **points** (*Union[geojson.feature.FeatureCollection, List]*) – To find the Voronoi polygons around. Points should be either FeatureCollection of points or list of points.
- **bbox** (*Optional[list]*) – A bounding box to clip.

Returns A GeoJSON Feature.

Return type geojson.feature.Feature

Example:

```
>>> from turfpy.transformation import voronoi
>>> points = [
... [-66.9703, 40.3183],
... [-63.7763, 40.4500],
... [-65.4196, 42.13985310302137],
... [-69.5813, 43.95405461286195],
... [-65.66337553550034, 55.97088945355232],
... [-60.280418548905, 56.240669185466146],
... [-68.5129561347689, 50.12984589640148],
... [-64.2393519226657, 59.66235385923687],
... ]
>>> bbox = [-70, 40, -60, 60]
>>> voronoi(points, bbox)
```

44.2.6 turfpy.__version__ module

Project version information.

CHAPTER

FORTYFIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

t

INDEX

Symbols

A

B

R

S

T

U

V