#Multi-dimensional arrays; create notebook through arrays tutorial docker container

In [87]:
```
import xarray as xr
```

In [88]:
```
open_dataset('/data/airtemp_AK.nc') #working with data from the EWF global reanaly
```

In [89]:
```
ds
```

Out[89]:
```
<xarray.Dataset>
Dimensions:    (latitude: 41, longitude: 94, time: 13361)
Coordinates:
  * latitude   (latitude) float32 75.0 74.25 73.5 72.75 72.0 71.25 70.5 ...
  * time       (time) datetime64[ns] 1979-01-01T06:00:00 1979-01-02T06:00:00
 ...
  * longitude  (longitude) float32 180.0 180.75 181.5 182.25 183.0 183.75 ...
Data variables:
    t2m        (time, latitude, longitude) float64 261.5 261.5 261.6 261.6 ...
Attributes:
    Conventions: CF-1.0
    history: 2015-10-19 21:11:12 GMT by grib_to_netcdf-1.13.1: grib_to_netcdf /
data/data04/scratch/netcdf-atls05-a562cefde8a29a7288fa0b8b7f9413f7-1hbxws.targe
t -o /data/data04/scratch/netcdf-atls05-a562cefde8a29a7288fa0b8b7f9413f7-y_OM5
J.nc -utime
```

In [90]:
```
ds.coords #don't need a () afterwards b/c it's not executing a function, just que
```

Out[90]:
```
Coordinates:
  * latitude   (latitude) float32 75.0 74.25 73.5 72.75 72.0 71.25 70.5 ...
  * time       (time) datetime64[ns] 1979-01-01T06:00:00 1979-01-02T06:00:00
 ...
  * longitude  (longitude) float32 180.0 180.75 181.5 182.25 183.0 183.75 ...
```

In [91]:
```
ds.attrs
```

Out[91]:
```
OrderedDict([('Conventions', 'CF-1.0'),
             ('history',
              '2015-10-19 21:11:12 GMT by grib_to_netcdf-1.13.1: grib_to_netcdf
 /data/data04/scratch/netcdf-atls05-a562cefde8a29a7288fa0b8b7f9413f7-1hbxws.tar
get -o /data/data04/scratch/netcdf-atls05-a562cefde8a29a7288fa0b8b7f9413f7-y_OM
5J.nc -utime')])
```

In [92]:
```
ds.dims
```

Out[92]:
```
Frozen(SortedKeysDict({'time': 13361, 'longitude': 94, 'latitude': 41}))
```

In [93]:
```
temperature = ds['t2m']
temperature #still has all of the same info as before
```

Out[93]:
```
<xarray.DataArray 't2m' (time: 13361, latitude: 41, longitude: 94)>
[51493294 values with dtype=float64]
Coordinates:
  * latitude   (latitude) float32 75.0 74.25 73.5 72.75 72.0 71.25 70.5 ...
  * time       (time) datetime64[ns] 1979-01-01T06:00:00 1979-01-02T06:00:00
 ...
  * longitude  (longitude) float32 180.0 180.75 181.5 182.25 183.0 183.75 ...
Attributes:
    long_name: 2 metre temperature
    units: K
```

In [94]:
```
ds['t2m'][0,0,0] #how we typically query in matlab or np array
```

Out[94]:
```
<xarray.DataArray 't2m' ()>
array(261.47730710087217)
Coordinates:
    latitude   float32 75.0
    time       datetime64[ns] 1979-01-01T06:00:00
    longitude  float32 180.0
Attributes:
    long_name: 2 metre temperature
    units: K
```

In [95]:
```
ds['t2m'][0,0,0].values #just get value
```

Out[95]:
```
array(261.47730710087217)
```

In [96]:
```
ds.coords['longitude'].min()
```

Out[96]:
```
<xarray.DataArray 'longitude' ()>
array(180.0)
```

In [97]:
```
ds['t2m'].loc['1979-06-01T06:00:00',:,:] #positional indexing using label instead
```

Out[97]:
```
<xarray.DataArray 't2m' (latitude: 41, longitude: 94)>
array([[ 272.69247415,  272.72074428,  272.74699513, ...,  266.94555924,
         267.30701313,  267.64423547],
       [ 272.87421074,  272.92469313,  272.97517551, ...,  269.51814167,
         269.43736985,  269.31621212],
       [ 273.04988945,  273.11248761,  273.17306647, ...,  270.04719708,
         269.88363415,  269.72007121],
       ...,
       [ 280.97562417,  280.67878774,  280.46474242, ...,  277.38935541,
         276.97539984,  276.57961793],
       [ 281.78940025,  281.41986917,  281.15736076, ...,  276.30297445,
         275.24688292,  274.79859932],
       [ 282.55673252,  282.15893132,  281.9004615 , ...,  275.77795763,
         274.31598771,  273.61933077]])
Coordinates:
  * latitude   (latitude) float32 75.0 74.25 73.5 72.75 72.0 71.25 70.5 ...
    time       datetime64[ns] 1979-06-01T06:00:00
  * longitude  (longitude) float32 180.0 180.75 181.5 182.25 183.0 183.75 ...
Attributes:
    long_name: 2 metre temperature
    units: K
```

In [98]:
```
 ds['t2m'].isel(time=0,latitude=0,longitude=0)#can go one step further; order of
```

Out[98]:
```
<xarray.DataArray 't2m' ()>
array(261.47730710087217)
Coordinates:
    latitude   float32 75.0
    time       datetime64[ns] 1979-01-01T06:00:00
    longitude  float32 180.0
Attributes:
    long_name: 2 metre temperature
    units: K
```

In [99]:
```
.sel(time='1979-06-01T06:00:00',longitude=180.0,latitude=75.0)#can combine best of
```

Out[99]:
```
<xarray.DataArray 't2m' ()>
array(272.6924741477606)
Coordinates:
    latitude   float32 75.0
    time       datetime64[ns] 1979-06-01T06:00:00
    longitude  float32 180.0
Attributes:
    long_name: 2 metre temperature
    units: K
```

In [100]:
```
time_series = ds['t2m'].sel(time=slice('1979-06-01T06:00:00','1980-06-01T06:00:00
#save that slice into a variable
time_series
```

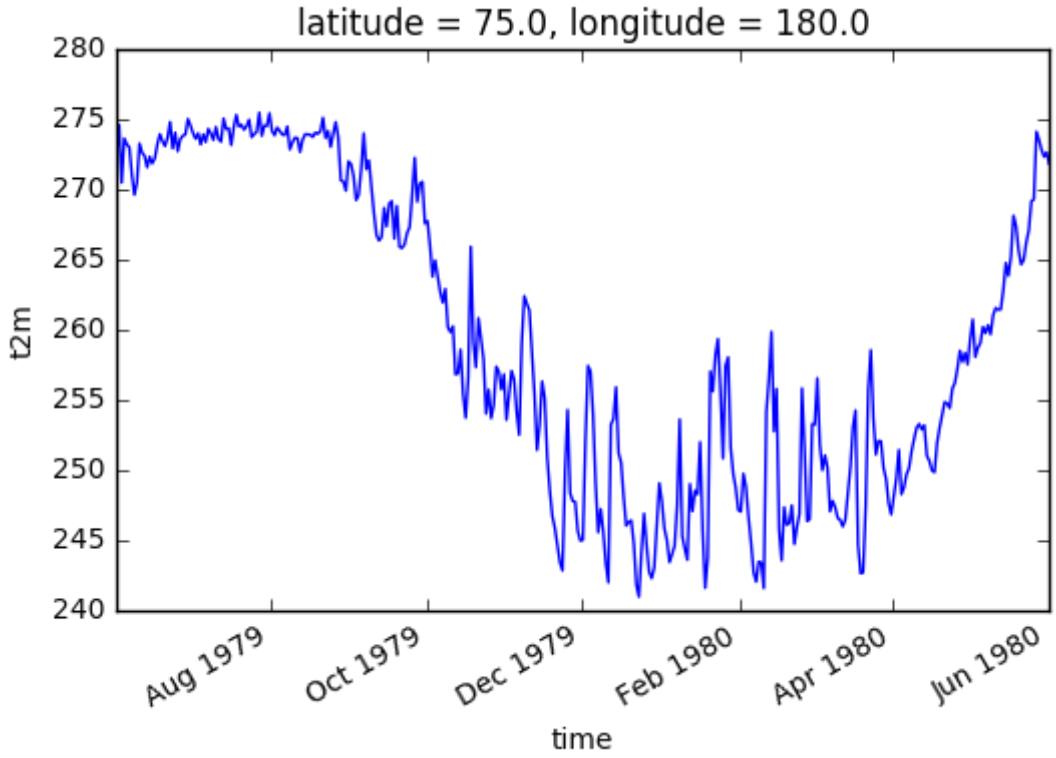
Out[100]:

```
<xarray.DataArray 't2m' (time: 367)>
array([ 272.69247415,  274.58253471,  270.47326842,  273.61933077,
        273.21143308,  272.98729129,  271.02857468,  269.60295208,
        270.49144208,  273.25989618,  272.60564444,  272.37544476,
        271.56166868,  272.32496237,  271.84033146,  272.20582394,
        273.20739449,  273.92626368,  273.43355559,  273.08017888,
        273.66779386,  274.79254144,  272.90853876,  274.03732493,
        272.70862851,  273.56480979,  273.81116384,  273.90001284,
        275.00658676,  274.53811021,  274.00299691,  273.60317641,
        273.98684254,  273.19124013,  273.90607073,  273.36893813,
        274.30185264,  273.962611  ,  273.5143274 ,  274.43512614,
        273.58096416,  273.39316968,  275.03687619,  274.29377546,
        274.34627714,  273.15085422,  274.42906825,  275.29736531,
        274.46339628,  274.59061189,  274.24531237,  274.48560853,
        274.95610437,  273.71423766,  273.93030227,  274.10194239,
        275.45890894,  273.81116384,  274.53407162,  274.51387866,
        275.41650374,  274.16857914,  273.84347257,  274.40281741,
        274.16655984,  273.90405143,  273.86164623,  274.47349275,
        272.83584413,  273.3366294 ,  273.67587104,  273.63346584,
        272.65612683,  273.52038529,  273.9181865 ,  273.91010932,
        273.88183918,  273.72635343,  274.03328634,  273.95655311,
        274.13425111,  275.09947435,  273.66173597,  274.1423283 ,
        273.01758072,  273.94645664,  274.78446426,  273.69202541,
        270.63481206,  270.54596306,  269.89574992,  271.98572073,
        271.79590696,  270.94376427,  269.22736312,  269.59689419,
        271.420318  ,  273.98280395,  271.43647236,  272.06043467,
        270.11181454,  268.30454509,  266.7375718 ,  266.34178989,
        266.61237548,  268.67205686,  267.3615341 ,  268.98302837,
        269.15264919,  266.48112128,  268.81138825,  265.89956418,
        265.80869589,  266.07524289,  266.88094178,  267.30499383,
        269.62516433,  272.24822914,  269.11024398,  270.38441942,
        270.53788588,  267.56750224,  267.73308447,  265.96014305,
        263.77728464,  264.93636024,  263.83382492,  262.68484579,
        261.93366788,  262.90898759,  260.16678434,  259.82350411,
        260.23947898,  256.80869597,  256.8854292 ,  258.5775988 ,
        255.39115055,  253.71513531,  256.57041911,  265.90764136,
        259.13492435,  257.31553913,  260.82305537,  259.36108545,
        257.90315412,  254.00995245,  255.72029571,  253.68282658,
        254.65208841,  257.35996364,  256.99649045,  255.72837289,
        256.82081174,  253.56974604,  255.35480323,  257.05908861,
        256.4411842 ,  254.13312947,  252.50355803,  259.03395958,
        262.38800936,  261.84078029,  261.37028444,  258.3554763 ,
        255.4153821 ,  251.42121565,  253.21636933,  256.3058914 ,
        255.13873862,  250.92850756,  248.56593186,  246.72635368,
        245.86613381,  244.6242671 ,  243.40461263,  242.83517131,
        249.81385646,  254.28255734,  248.35188654,  247.7905224 ,
        247.72388565,  245.64199202,  244.96552803,  245.01802972,
        252.12796907,  257.4205425 ,  257.0187027 ,  253.93927711,
        249.03238911,  245.55314302,  247.21300389,  245.60160611,
        243.35211095,  241.98504792,  253.23858158,  253.62224772,
        255.89597442,  251.21726681,  250.53676424,  248.07524306,
        246.02969675,  246.28210868,  246.40932429,  244.74340553,
        241.80936921,  240.967323  ,  244.1335783 ,  246.88789732,
        244.63234428,  242.74632231,  242.28188435,  243.03508156,
        246.12662293,  249.05863995,  247.76225226,  245.82372861,
        244.97764381,  243.42278629,  244.02453634,  244.58993908,
        247.46339653,  253.60609335,  245.2906346 ,  244.39406741,
```

```
        243.59442641,  249.00411898,  247.06357603,  248.52958454,
        248.25899895,  251.98661839,  245.69853229,  241.5933046 ,
        243.78424018,  257.02476059,  255.60317658,  258.17172041,
        259.3368539 ,  255.58298362,  250.82754279,  257.4205425 ,
        258.03036973,  251.60901013,  249.72298816,  248.70930184,
        247.14434785,  247.029248  ,  249.73510394,  248.77997718,
        246.63750468,  244.80600369,  242.71603288,  242.02947242,
        243.47326868,  243.38643897,  241.57311164,  254.15130313,
        256.47753151,  259.84369706,  252.74587348,  255.78087458,
        245.72276383,  243.5540405 ,  247.3442581 ,  246.09229491,
        246.21749123,  247.47955089,  244.71513539,  245.94488634,
        246.87578155,  255.82126049,  251.43938931,  246.35682261,
        246.48403823,  253.26685172,  253.23858158,  256.53811038,
        251.80488179,  249.98145798,  251.0436074 ,  250.15511739,
        247.04338307,  247.79658028,  247.34829669,  246.57894511,
        246.43355584,  245.97315647,  246.46384527,  248.3680409 ,
        250.23386992,  253.06896076,  254.24217143,  244.66061442,
        242.62920317,  242.69987851,  247.47147371,  255.72837289,
        258.54327078,  253.70907742,  251.10014767,  252.07950598,
        252.03508148,  250.14098233,  249.3494185 ,  247.63099806,
        246.83337634,  248.07322376,  249.36355357,  251.43131213,
        248.26909542,  248.72949479,  249.67856366,  250.13694373,
        251.33640524,  252.22691455,  253.03059414,  253.28098678,
        252.88116628,  253.1860799 ,  251.01331797,  250.6437869 ,
        249.97136151,  249.83404941,  251.91998164,  252.99828542,
        253.93927711,  254.83988289,  254.73487952,  254.40977295,
        255.79299035,  256.20896522,  257.23072872,  258.49682699,
        257.69718598,  258.34537983,  257.52554586,  259.31262236,
        260.72410989,  258.05056269,  258.70683371,  259.04809465,
        260.19303518,  259.75080947,  260.31217361,  259.66801836,
        261.01690773,  261.54394385,  261.41672824,  261.44297908,
        262.82619648,  264.75462365,  263.8419021 ,  265.19483006,
        268.11876991,  267.37163058,  265.64917154,  264.6435624 ,
        264.96463038,  266.20649709,  267.07277485,  269.13649482,
        269.28390339,  274.10598098,  273.52240459,  272.81767047,
        272.31688519,  272.62785669,  271.81004203])
Coordinates:
    latitude   float32 75.0
  * time       (time) datetime64[ns] 1979-06-01T06:00:00 1979-06-02T06:00:00
 ...
    longitude  float32 180.0
Attributes:
    long_name: 2 metre temperature
    units: K
```

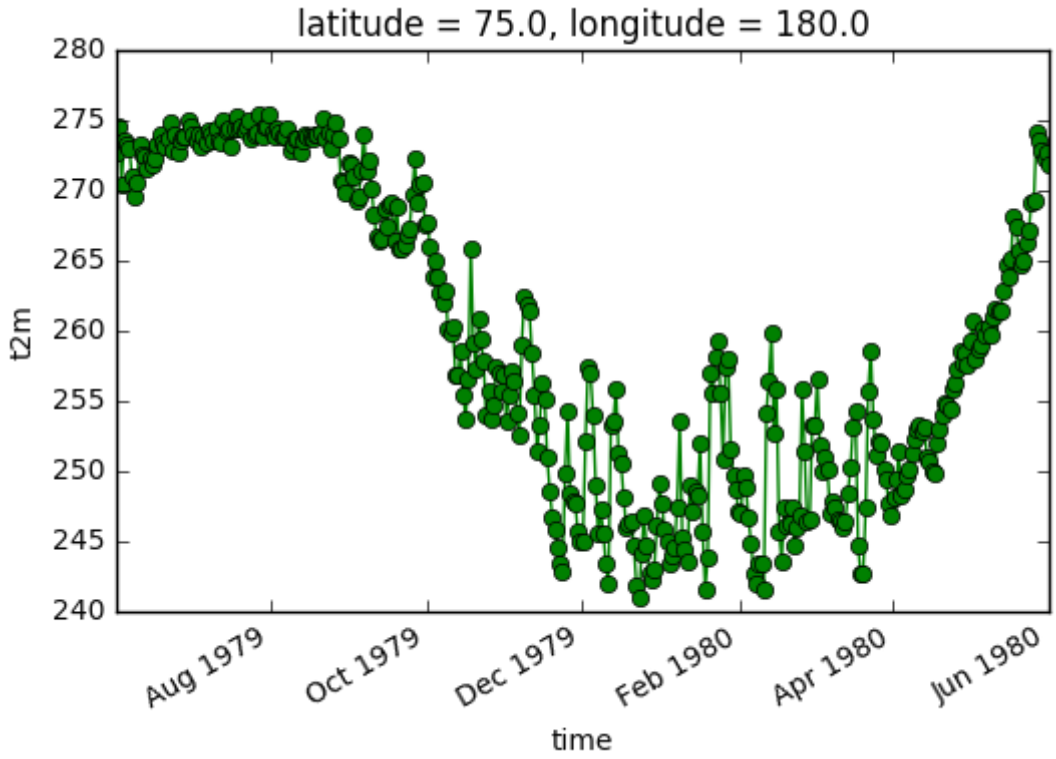
In [101]:
```
%matplotlib inline
time_series.plot() #since our data is a slice at a point, can do this quick plot
```

Out[101]: [<matplotlib.lines.Line2D at 0x7f8ce069bba8>]



In [102]:
```
time_series.plot.line(color='green',marker='o')
```

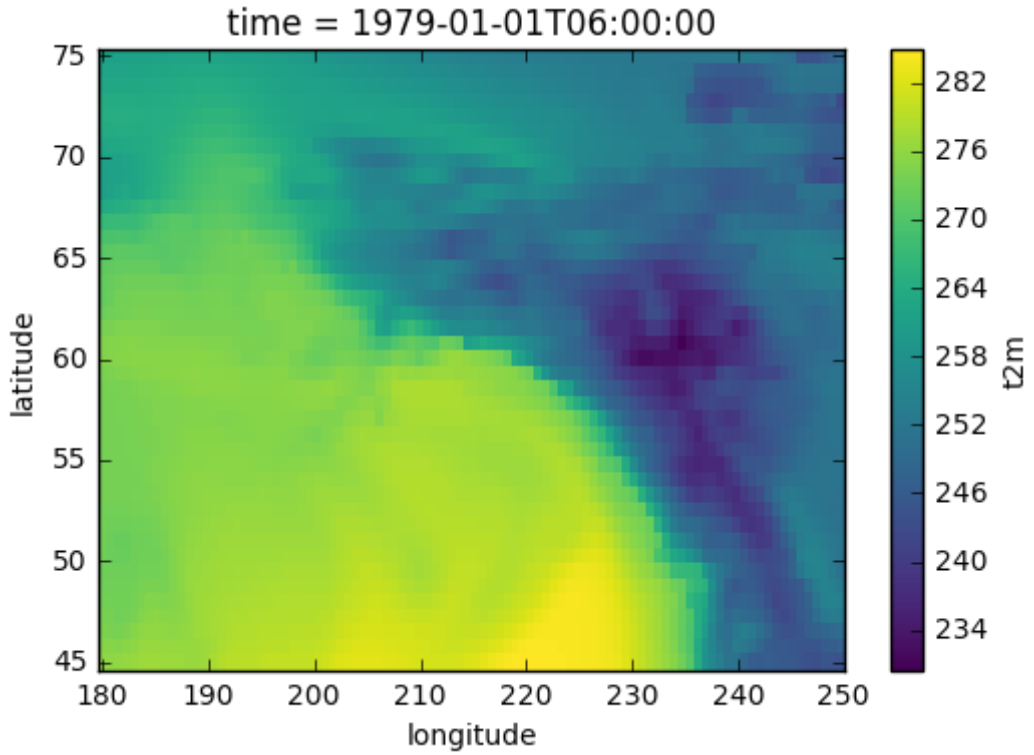
Out[102]: [<matplotlib.lines.Line2D at 0x7f8ce0b76240>]



In [103]:
```
map_data = ds['t2m'].sel(time='1979-01-01T06:00:00') #because the other dims aren
```

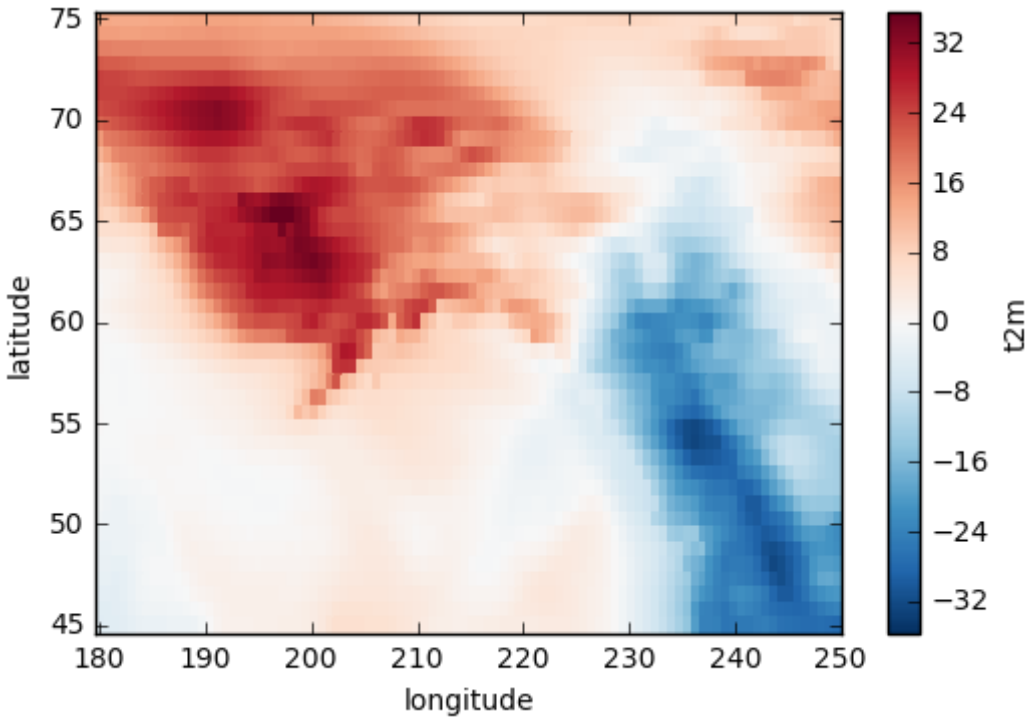
In [104]:
```
map_data.plot()
```

Out[104]:
```
<matplotlib.collections.QuadMesh at 0x7f8ce07f1c18>
```



In [105]:
```
temp1 = ds['t2m'].sel(time='1979-01-01T06:00:00')
temp2 = ds['t2m'].sel(time='1980-01-01T06:00:00')
delta = temp1-temp2 #create var with differences in temp b/w the 2 years
delta.plot()
```

Out[105]:
```
<matplotlib.collections.QuadMesh at 0x7f8ce0603e80>
```



In [106]:
```
import xarray.ufuncs as xu
import matplotlib.pyplot as plt
```

In [107]:
```
#Play with sqrt function which are in xu -- use wind data since that allows us to
wind = xr.open_mfdataset('/data/*wind_AK.nc').sel(time="1984-01-01T06:00:00") #th
wind #have both v10 and u10 variables under data :-)
```

Out[107]:
```
<xarray.Dataset>
Dimensions:    (latitude: 41, longitude: 94)
Coordinates:
  * latitude   (latitude) float32 75.0 74.25 73.5 72.75 72.0 71.25 70.5 ...
    time       datetime64[ns] 1984-01-01T06:00:00
  * longitude  (longitude) float32 180.0 180.75 181.5 182.25 183.0 183.75 ...
Data variables:
    v10        (latitude, longitude) float64 1.553 1.169 0.7863 0.4024 ...
    u10        (latitude, longitude) float64 6.798 6.944 7.09 7.236 7.311 ...
```

In [108]:
```
windspeed = xu.sqrt(wind.u10**2 + wind.v10**2) #pythag theorem
```

In [109]:
```
windspeed.plot(cmap=plt.cm.Blues)
plt.title('ECMWF wind speed and direction, June 1, 1984')
plt.ylabel('latitude')
plt.xlabel('longitude')
x = windspeed.coords['longitude'].values
y = windspeed.coords['latitude'].values
plt.quiver(x, y, wind.u10.values, wind.v10.values)
```

Out[109]: <matplotlib.quiver.Quiver at 0x7f8ce278d5f8>