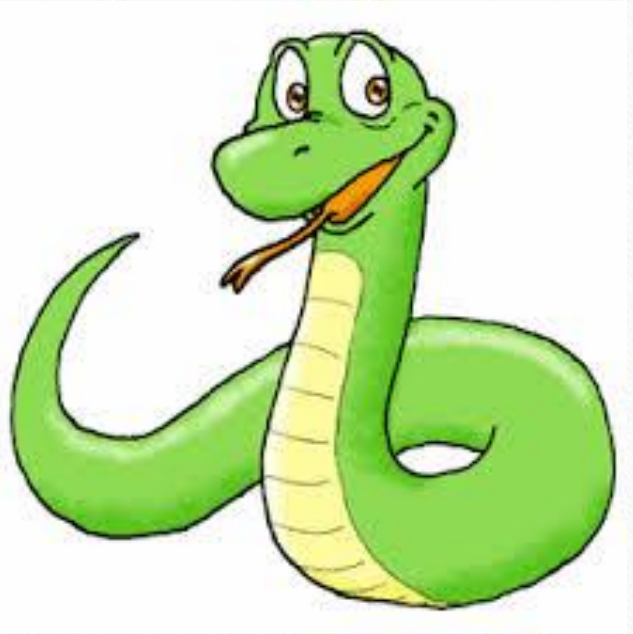
APPLICATION OF GIS WITH PYTHON

9. Working with images



https://pcjericks.github.io/py-gdalogr-cookbook/

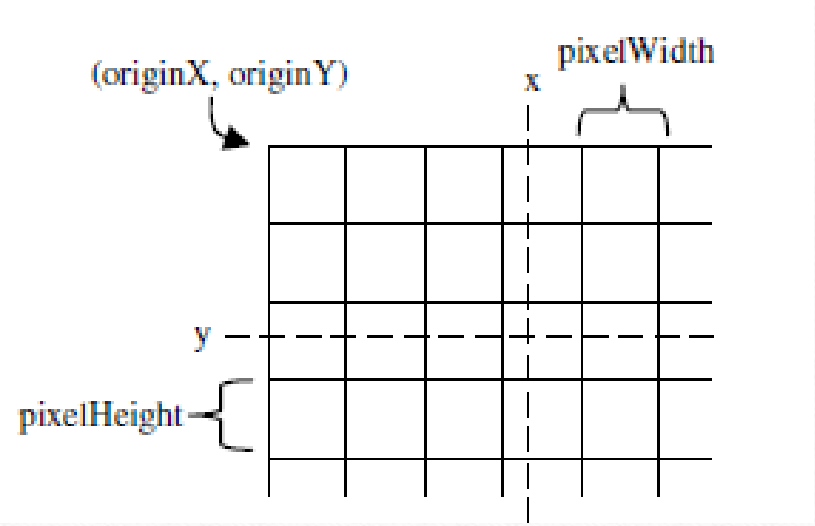
https://www.python.org/

Raster Data

```
Driver: GTiff/GeoTIFF
Files: world.tif
Size is 2048, 1024
Coordinate System is:
        GEOGCS["WGS 84",
                DATUM["WGS_1984",
                SPHEROID["WGS 84",6378137,298.257223563,
                    AUTHORITY["EPSG","7030"]],
                    AUTHORITY["EPSG","6326"]],
                PRIMEM["Greenwich",0],
                UNIT["degree",0.0174532925199433],
        AUTHORITY["EPSG","4326"]]
Origin = (-180.000000000000000,90.000000000000000)
Pixel Size = (0.175781250000000,-0.175781250000000)
```

```
Metadata:
AREA_OR_POINT=Area
Image Structure Metadata:
INTERLEAVE=BAND
Corner Coordinates:
Upper Left (-180.0000000, 90.0000000) (180d 0' 0.00"W, 90d 0' 0.00"N)
Lower Left (-180.0000000, -90.0000000) (180d 0' 0.00"W, 90d 0' 0.00"S)
Upper Right ( 180.0000000, 90.0000000) (180d 0' 0.00"E, 90d 0' 0.00"N)
Lower Right ( 180.0000000, -90.0000000) (180d 0' 0.00"E, 90d 0' 0.00"S)
Center ( 0.0000000, 0.0000000) ( 0d 0' 0.01"E, 0d 0' 0.01"N)
Band 1 Block=256x256 Type=Byte, ColorInterp=Red
Overviews: 1024x512, 512x256, 256x128, 128x64, 64x32, 32x16, 16x8
Band 2 Block=256x256 Type=Byte, ColorInterp=Green
Overviews: 1024x512, 512x256, 256x128, 128x64, 64x32, 32x16, 16x8
Band 3 Block=256x256 Type=Byte, ColorInterp=Blue
Overviews: 1024x512, 512x256, 256x128, 128x64, 64x32, 32x16, 16x8
```

Raster Data



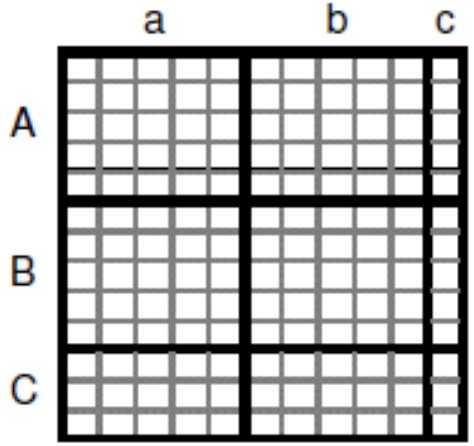


Figure 2. An image with 13 rows, 11 columns, and a blocksize of 5.

Reading raster data

- Need to import two modules in order to work with raster
- The gdal module does the hard work, but the gdalconst module includes the constants that are required for some things

```
import gdal
from gdalconst import *
```

Reading raster data

- The supported image formats are described at http://www.gdal.org/formats_list.html and there is an associated driver for each one. Or type in command prompt

  gdalinfo --formats

- In case of GDAL get a driver before we open a file. And need to register a driver before use.

```
driver = gdal.GetDriverByName('HFA')
driver.Register()
```

*In osgeo distribution raster may open and read without driver

Reading raster data

```
from osgeo import gdal
import os
os.chdir("E:\\WRC\\ospy_data2")
ds = gdal.Open('aster.img')
print('File list:', ds.GetFileList())
print('Width:', ds.RasterXSize)
print('Height:', ds.RasterYSize)
print('Coordinate system:', ds.GetProjection())
gt = ds.GetGeoTransform() # captures origin and pixel size
print('Origin:', (gt[0], gt[3]))
print('Pixel size:', (gt[1], gt[5]))
print('Upper Left Corner:', gdal.ApplyGeoTransform(gt,0,0))
print('Upper Right Corner:', gdal.ApplyGeoTransform(gt,ds.RasterXSize,0))
print('Lower Left Corner:', gdal.ApplyGeoTransform(gt,0,ds.RasterYSize))
print('Lower Right Corner:',gdal.ApplyGeoTransform(gt,ds.RasterXSize,ds.RasterYSize))
print('Center:', gdal.ApplyGeoTransform(gt,ds.RasterXSize/2,ds.RasterYSize/2))
```

```
print('Metadata:', ds.GetMetadata())
print('Image Structure Metadata:', ds.GetMetadata('IMAGE_STRUCTURE'))
print('Number of bands:', ds.RasterCount)
for i in range(1, ds.RasterCount+1):
    band = ds.GetRasterBand(i) # in GDAL, band are indexed starting at 1!
    interp = band.GetColorInterpretation()
    interp_name = gdal.GetColorInterpretationName(interp)
    (w,h)=band.GetBlockSize()
    print('Band %d, block size %dx%d, color interp %s' % (i,w,h,interp_name))
    ovr_count = band.GetOverviewCount()
    for j in range(ovr_count):
        ovr_band = band.GetOverview(j) # but overview bands starting at 0
        print(' Overview %d: %dx%d'%(j, ovr_band.XSize, ovr_band.YSize))
```

Reading raster data

```
import os, gdal
os.chdir(r'E:\WRC\2015winter\Gis With Python\labs\pyFiles\gggddaall')
#information about 288316a.tif
ds=gdal.Open('288316a/288316a.tif')

#Files : list of files. Main file + potential additional files
print('File list:', ds.GetFileList())
#raster width in pixels
print('Width:', ds.RasterXSize)
#raster height in pixels
print('Height:', ds.RasterYSize)
#spatial reference system in WKT format
print('Coordinate system:', ds.GetProjection())
gt = ds.GetGeoTransform() # captures origin and pixel size click here
#corner and centre coordinates
print('Origin:', (gt[0], gt[3]))
print('Pixel size:', (gt[1], gt[5]))
print('Upper Left Corner:', gdal.ApplyGeoTransform(gt,0,0))
print('Upper Right Corner:', gdal.ApplyGeoTransform(gt,ds.RasterXSize,0))
print('Lower Left Corner:', gdal.ApplyGeoTransform(gt,0,ds.RasterYSize))
print('Lower Right Corner:',gdal.ApplyGeoTransform(gt,ds.RasterXSize,ds.RasterYSize))
print('Center:', gdal.ApplyGeoTransform(gt,ds.RasterXSize/2,ds.RasterYSize/2))
```

```
#metadata of raster file, a list of KEY=VALUE pairs, depending on the format and data
print('Metadata:', ds.GetMetadata())
#like arrangement of pixels; INTERLEAVE=BAND:first all pixels for first band then others
print('Image Structure Metadata:', ds.GetMetadata('IMAGE_STRUCTURE'))
print('Number of bands:', ds.RasterCount)

#looping for each band
for i in range(1, ds.RasterCount+1):
    band = ds.GetRasterBand(i) # in GDAL, band are indexed starting at 1!
    interp = band.GetColorInterpretation()#display color
    interp_name = gdal.GetColorInterpretationName(interp)
    (w,h)=band.GetBlockSize()#block size:lines(col,row),square,tiles
    print('Band %d, block size %dx%d, color interp %s' % (i,w,h,interp_name))
    #reduced resolution size of full resolution
    ovr_count = band.GetOverviewCount()
    for j in range(ovr_count):
        ovr_band = band.GetOverview(j) # but overview bands starting at 0
        print(' Overview %d: %dx%d'%(j, ovr_band.XSize, ovr_band.YSize))
```

Reading raster data

```
('File list:', ['288316a/288316a.tif', '288316a/288316a.aux', '288316a\\288316a
.rrd', '288316a/288316a.tfw'])
('Width:', 3893)
('Height:', 4382)
('Coordinate system:', '')
('Origin:', (475432.07544513594, 3125603.476104019))
('Pixel size:', (3.161612083157206, -3.171123337312481))
('Upper Left Corner:', [475432.07544513594, 3125603.476104019])
('Upper Right Corner:', [487740.2312848669, 3125606.7197328974])
('Lower Left Corner:', [475432.50026312115, 3111707.613639916])
('Lower Right Corner:', [487740.65610285214, 3111710.857268794])
('Center:', [481584.7849679525, 3118657.166269809])
('Metadata:', {'TIFFTAG_XRESOLUTION': '200', 'TIFFTAG_IMAGEDESCRIPTION': '', 'T
IFFTAG_DATETIME': '2008:02:29 14:23:53', 'TIFFTAG_RESOLUTIONUNIT': '2 (pixels/i
nch)', 'TIFFTAG_SOFTWARE': 'IMAGINE TIFF Support\nCopyright 1991 - 1999 by ERDA
S, Inc. All Rights Reserved\n@(#)$RCSfile: etif.c $ $Revision: 1.9.1.3 $ $Date:
2002/07/29 15:39:06EDT $', 'TIFFTAG_YRESOLUTION': '200'})
('Image Structure Metadata:', {'INTERLEAVE': 'BAND'})
('Number of bands:', 1)
Band 1, block size 3893x2, color interp Palette
 Overview 0: 974x1096
 Overview 1: 487x548
 Overview 2: 244x274
 Overview 3: 122x137
 Overview 4: 61x69
```

Opening a raster data set

- Once the driver has been registered, the `Open(<filename>, <GDALAccess>)` method can be used to return a Dataset object.

```
fn = 'aster.img'
ds = gdal.Open(fn, GA_ReadOnly)
if ds is None:
    print 'Could not open ' + fn
    sys.exit(1)
```

Getting image dimensions

- Dataset objects have properties corresponding to numbers of rows, columns and bands in the data set

```
cols = ds.RasterXSize
rows = ds.RasterYSize
bands = ds.RasterCount
```

Notice no parentheses

Getting georeference info

- GeoTransforms are lists of information used to georeference an image

```
gt = ds.GetGeoTransform()
```

```
gt[0] # top left x
gt[1] # w-e pixel resolution; pixel width
gt[2] # rotation, 0 if image is "north up"
gt[3] # top left y
gt[4] # rotation, 0 if image is "north up"
gt[5] # n-s pixel resolution; pixel height
```

Coordinates are for top left corners of pixels (unlike Imagine, which uses centers)

Getting georeference info

- This is exactly the computation done by gdal.ApplyGeoTransform :

  [X,Y]=gdal.ApplyGeoTransform(gt,row,col)

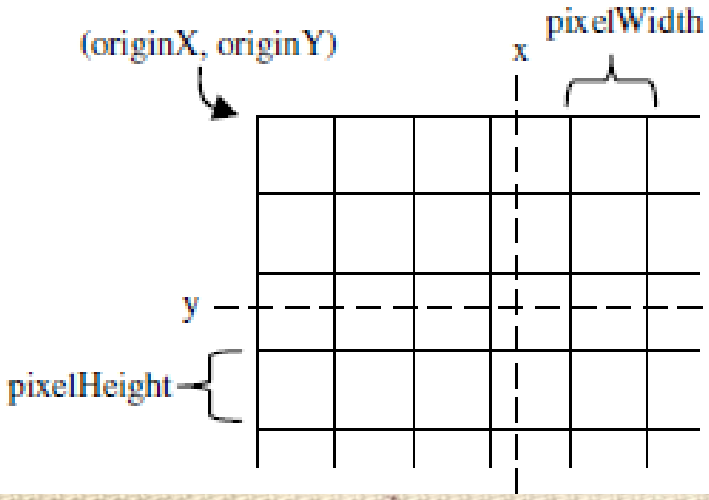
```
X = gt[0] + col * gt[1] + row * gt[2]
Y = gt[3] + col * gt[4] + row * gt[5]
```

- described as an affine transformation from the coordinates in the pixel space (col,row) to the coordinates of the projected space (X,Y), with col and row starting from 0 for the upper-left pixel
- Where, gt = ds.GetGeoTransform()

Computing pixel offsets

- Need to get pixel offsets from the upper left corner for specific coordinates x,y

```
xOffset = int((x - originX) / pixelWidth)
yOffset = int((y - originY) / pixelHeight)
```




```
(x - originX) / pixelWidth ~= 3.25
(y - originY) / pixelHeight ~= 2.5
```

Getting individual pixel values

- Get the Band object by passing the band index (1-based) to the Dataset's **GetRasterBand(<index>)** method

```
band = ds.GetRasterBand(1)
```

- Read the data into a 2D Numeric array with **ReadAsArray(<xoff>, <yoff>,<xsize>, <ysize>)**

```
data = band.ReadAsArray(xOffset, yOffset, 1, 1)
```

Getting individual pixel values

- Even though we only read one pixel value, it is in a two-dimensional array
- Since we read one pixel in each direction, the array is of size 1x1
- Need to specify both offsets, which are 0 in this case

```
value = data[0,0]
print value
```

Reading an entire image at once

- Use 0 offsets and pass the numbers of rows and columns to the **ReadAsArray()**

```
data = band.ReadAsArray(0, 0, cols, rows)
```

- Read individual pixels using [yoff, xoff] (math matrix notation is [row,col], not [x,y])
- To read the pixel in the 95th column and 43rd row:

```
value = data[42, 94]
```

Memory management

- Set variables to None
- Especially important if you created large arrays with **ReadAsArray()**

```
# close dataset
band = None
dataset = None
```

Example exercise

```
# script to get pixel values at a set of coordinates
# by reading in one pixel at a time
# Took 0.311999797821 seconds on my machine
import os, gdal, sys, time
# start timing
startTime = time.time()

# coordinates to get pixel values for
xValues = [476520.0, 477524.0, 476503.0]
yValues = [3112976.0, 3112827.0, 3112114.0]
#set directory
os.chdir(r'E:\WRC\2015winter\Gis With Python\lab
ds=gdal.Open('288316a/288316a.tif')
if ds is None:
    print 'Could not open image'
    sys.exit(1)

# get image size
rows = ds.RasterYSize
cols = ds.RasterXSize
bands = ds.RasterCount

# get georeference info
transform = ds.GetGeoTransform()
xOrigin = transform[0]
yOrigin = transform[3]
pixelWidth = transform[1]
pixelHeight = transform[5]
```

```
x          y          ofX  ofY  PxlVal
476520.0 3112976.0 344 3982 130
477524.0 3112827.0 661 4029 252
476503.0 3112114.0 338 4253 245
The script took 0.0809998512268 sec
```

```
# loop through the coordinates
for i in range(3):
    # get x,y
    x = xValues[i]
    y = yValues[i]
    # compute pixel offset
    xOffset = int((x - xOrigin) / pixelWidth)
    yOffset = int((y - yOrigin) / pixelHeight)
    # create a string to print out
    s = str(x) + ' ' + str(y) + ' ' + str(xOffset) + ' ' + str(yOffset) + ' '
    # loop through the bands
    for j in range(bands):
        band = ds.GetRasterBand(j+1) # 1-based index
        # read data and add the value to the string
        data = band.ReadAsArray(xOffset, yOffset, 1, 1)
        value = data[0,0]
        s = s + str(value) + ' '
    # print out the data string
    print s
# figure out how long the script took to run
endTime = time.time()
print 'The script took ' + str(endTime - startTime) + ' seconds'
```

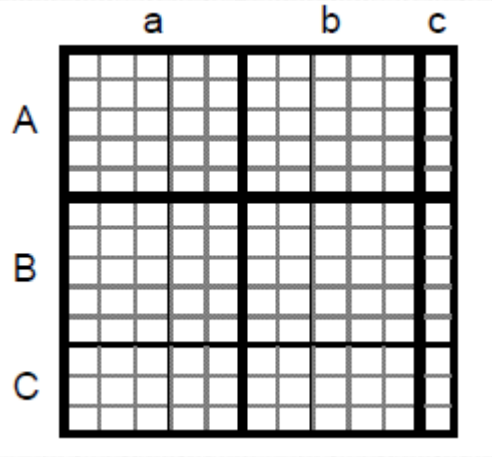
Example exercise

```
# script to get pixel values at a set of coordinates
# by reading in one pixel at a time
# Took 0.311999797821 seconds on my machine
import os, gdal, sys, time
# start timing
startTime = time.time()
# coordinates to get pixel values for
xValues = [476520.0, 477524.0, 476503.0]
yValues = [3112976.0, 3112827.0, 3112114.0]
#set directory
os.chdir(r'E:\WRC\gggddaall')
ds=gdal.Open('288316a/288316a.tif')
if ds is None:
    print 'Could not open image'
    sys.exit(1)
# get image size
rows = ds.RasterYSize
cols = ds.RasterXSize
bands = ds.RasterCount
# get georeference info
transform = ds.GetGeoTransform()
xOrigin = transform[0]
yOrigin = transform[3]
pixelWidth = transform[1]
pixelHeight = transform[5]
# loop through the coordinates
for i in range(3):
    # get x,y
    x = xValues[i]
    y = yValues[i]
    # compute pixel offset
    xOffset = int((x - xOrigin) / pixelWidth)
    yOffset = int((y - yOrigin) / pixelHeight)
    # create a string to print out
    s = str(x) + ' ' + str(y) + ' ' + str(xOffset) + ' ' + str(yOffset) + '
'
    # loop through the bands
    for j in range(bands):
        band = ds.GetRasterBand(j+1) # 1-based index
        # read data and add the value to the string
        data = band.ReadAsArray(xOffset, yOffset, 1, 1)
        value = data[0,0]
        s = s + str(value) + ' '
    # print out the data string
    print s
# figure out how long the script took to run
endTime = time.time()
print 'The script took ' + str(endTime - startTime) + ' seconds'
```

```
x          y          ofX  ofY  PxlVal
476520.0 3112976.0 344 3982 130
477524.0 3112827.0 661 4029 252
476503.0 3112114.0 338 4253 245
The script took 0.0809998512268 sec
```

Reading block by block

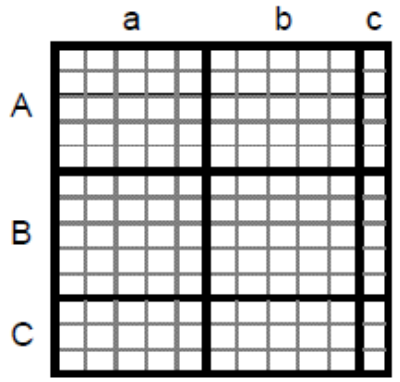
- The most efficient way to read data
- Use one loop for the rows and one for the columns
- Need to check that there is an entire block in both directions



Reading block by block

```
rows = 13, cols = 11
range(0,13,5) & range(0,11,5) both return [0, 5, 10]

xBSize = 5
yBSize = 5
for i in range(0, rows, yBSize):
  if i + yBSize < rows:
      numRows = yBSize
  else:
      numRows = rows - i
  for j in range(0, cols, xBSize):
      if j + xBSize < cols:
            numCols = xBSize
      else:
            numCols = cols - j
      data = band.ReadAsArray(j, i, numCols, numRows)
      # do something with the data here, before
      # reading the next block
```



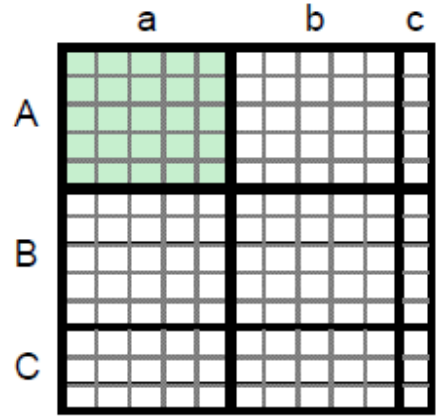
Reading block by block

```
rows = 13, cols = 11, xBSize = 5, yBSize = 5
for i in range(0, rows, yBSize):
  if i + yBSize < rows:
    numRows = yBSize
  else:
    numRows = rows - i
  for j in range(0, cols, xBSize):
    if j + xBSize < cols:
      numCols = xBSize
    else:
      numCols = cols - j
    data = band.ReadAsArray(j, i, numCols, numRows)
```

i = [**0**, 5, 10]
0 + 5 < 13, so numRows = 5

j = [**0**, 5, 10]
0 + 5 < 11, so numCols = 5

ReadAsArray(0, 0, 5, 5)



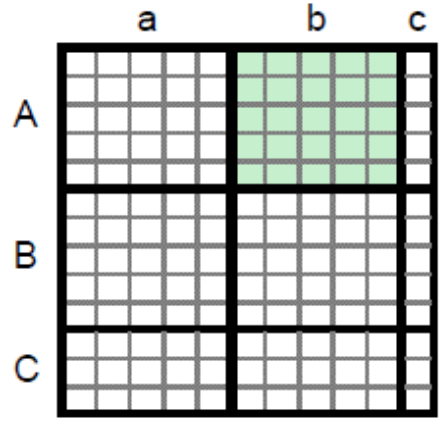
Reading block by block

```
rows = 13, cols = 11, xBSize = 5, yBSize = 5
for i in range(0, rows, yBSize):
  if i + yBSize < rows:
    numRows = yBSize
  else:
    numRows = rows - i
  for j in range(0, cols, xBSize):
    if j + xBSize < cols:
      numCols = xBSize
    else:
      numCols = cols - j
    data = band.ReadAsArray(j, i, numCols, numRows)
```

i = [**0**, 5, 10]
0 + 5 < 13, so numRows = 5

j = [0, **5**, 10]
5 + 5 < 11, so numCols = 5

ReadAsArray(5, 0, 5, 5)



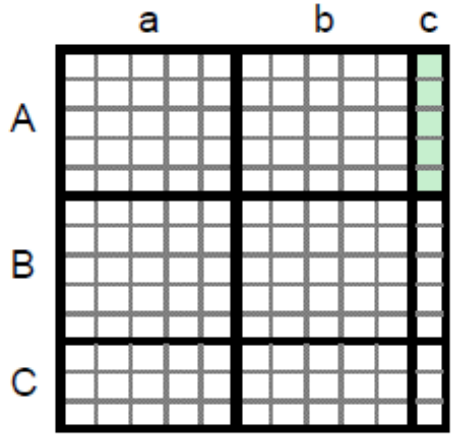
Reading block by block

```
rows = 13, cols = 11, xBSize = 5, yBSize = 5
for i in range(0, rows, yBSize):
  if i + yBSize < rows:
    numRows = yBSize
  else:
    numRows = rows - i
  for j in range(0, cols, xBSize):
    if j + xBSize < cols:
      numCols = xBSize
    else:
      numCols = cols - j
    data = band.ReadAsArray(j, i, numCols, numRows)
```

i = [**0**, 5, 10]
0 + 5 < 13, so numRows = 5

j = [0, 5, **10**]
10 + 5 > 11, so numCols = 11 – 10 = 1

ReadAsArray(10, 0, 1, 5)



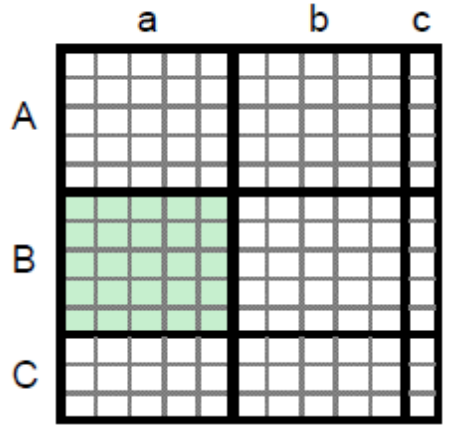
Reading block by block

```
rows = 13, cols = 11, xBSize = 5, yBSize = 5
for i in range(0, rows, yBSize):
  if i + yBSize < rows:
    numRows = yBSize
  else:
    numRows = rows - i
  for j in range(0, cols, xBSize):
    if j + xBSize < cols:
      numCols = xBSize
    else:
      numCols = cols - j
    data = band.ReadAsArray(j, i, numCols, numRows)
```

i = [0, **5**, 10]
5 + 5 < 13, so numRows = 5

j = [**0**, 5, 10]
0 + 5 < 11, so numCols = 5

ReadAsArray(0, 5, 5, 5)



Writing raster data

- Need the appropriate driver
- Fetch the driver using the GetDriverByName() method
- The Create() method requires us to pass in the filename for the new image, the numbers of columns, rows and bands, and a constant specifying the data type

```
import gdal, osr, os
```

```
driver = gdal.GetDriverByName('GTiff')
outRaster = driver.Create('newRasterfn.tif', cols, rows, 1, gdal.GDT_Float32)
```

Creating raster data

- If we want to write our data to a same file type as source/input raster dataset can fetch the driver from source then create new raster file of the same type.

```
driver = inDataset.GetDriver()
outRaster = driver.Create('newRasterfn.tif', cols, rows, 1, gdal.GDT_Float32)
```

Setting raster data

- To georeference the new image

```
outRaster.SetGeoTransform((originX, pixelWidth, 0, originY, 0, pixelHeight))
```

- To set projection

```
outRasterSRS = osr.SpatialReference()
outRasterSRS. spatialRef.ImportFromEPSG(26912)
outRaster.SetProjection(outRasterSRS.ExportToWkt())
```

Setting raster data

- To georeference the new image as source dataset
- To set projection as source dataset

```
outDataset.SetGeoTransform(inDataset.GetGeoTransform())
outDataset.SetProjection(inDataset.GetProjection())
```

Writing raster data

- Then get the band object from output raster dataset and write array data into the band.

```
outBand = outDataset.GetRasterBand(1)
outBand.WriteArray(array, 0, 0)
```

- The first parameter is the array of data to write into the band, the second is the X offset to start writing at, and the third is the Y offset to start writing at

```
pt to write NDVI raster after processing input raster
 lots of seconds on my machine
 os, gdal, sys, time
t timing
ime = time.time()

irectory
ir(r'E:\WRC\2015winter\Gis With Python\labs\pyFiles\gggddaall\osp
dal.Open('aster.img')
s is None:
int 'Could not open image'
s.exit(1)
aster size
 inDs.RasterYSize
 inDs.RasterXSize
driver from source raster file
 = inDs.GetDriver()
te a new dataset object from the driver
= driver.Create('NDVIaster.img', cols, rows, 1, gdal.GDT_Float32)

georeferencing and projection from source raster file
SetGeoTransform(inDs.GetGeoTransform())
SetProjection(inDs.GetProjection())

oand object from source raster dataset
3=inDs.GetRasterBand(3)
2=inDs.GetRasterBand(2)
oand object from new raster dataset
d = outDs.GetRasterBand(1)
```

```
blockSize = 64
for i in range(0, rows, blockSize):
    if i + blockSize < rows:
        numRows = blockSize
    else:
        numRows = rows - i
    for j in range(0, cols, blockSize):
        if j + blockSize < cols:
            numCols = blockSize
        else:
            numCols = (cols-j)
        data3 = inBand3.ReadAsArray(j, i, numCols, numRows)
        data2 = inBand2.ReadAsArray(j, i, numCols, numRows)
        # do something with the data here, before reading the
        dataD=(data3 + data2)
        for m in range(numRows):
            for n in range(numCols):
                if dataD[m,n]==0:
                    dataD[m,n]=1

        ndvi = (data3 - data2) /dataD
        outBand.WriteArray(ndvi, j, i)

# memory management
inBand2=None
inBand3=None
outBand=None
outDs=None
inDs=None
endTime = time.time()
print 'The script took ' + str(endTime - startTime) + ' second
```

Writing raster data

```
# script to write output raster after processing input raster
# Took lots of seconds on my machine
import os, gdal, sys, time
# start timing
startTime = time.time()
#set directory
os.chdir(r'E:\WRC\2015winter\Gis With Python')
inDs=gdal.Open('aster.img')
if inDs is None:
    print 'Could not open image'
    sys.exit(1)
#get raster size
rows = inDs.RasterYSize
cols = inDs.RasterXSize
# get driver from source raster file
driver = inDs.GetDriver()
# create a new dataset object from the driver
outDs = driver.Create('NDVIaster.img', cols, rows, 1, gdal.GDT_Float32)
# get georeferencing and projection from source raster file
outDs.SetGeoTransform(inDs.GetGeoTransform())
outDs.SetProjection(inDs.GetProjection())
# get band object from source raster dataset
inBand3=inDs.GetRasterBand(3)
inBand2=inDs.GetRasterBand(2)
# get band object from new raster dataset
outBand = outDs.GetRasterBand(1)
blockSize = 64
for i in range(0, rows, blockSize):
    if i + blockSize < rows:
        numRows = blockSize
    else:
        numRows = rows - i
    for j in range(0, cols, blockSize):
        if j + blockSize < cols:
            numCols = blockSize
        else:
            numCols = (cols-j)
        data3 = inBand3.ReadAsArray(j, i, numCols, numRows)
        data2 = inBand2.ReadAsArray(j, i, numCols, numRows)
# do something with the data here, before reading the next bl
        dataD=(data3 + data2)
        for m in range(numRows):
            for n in range(numCols):
                if dataD[m,n]==0:
                    dataD[m,n]=1

        ndvi = (data3 - data2) /dataD
        outBand.WriteArray(ndvi, j, i)
```

```
# memory management
inBand2=None
inBand3=None
outBand=None
outDs=None
inDs=None
endTime = time.time()
print 'The script took ' + str(endTime - startTime) + ' seconds.'
```

Image is read blockwise and written blockwise. Can read and write entire raster file. Check the time taken to process trying entire raster reading and writing.