```python
#PSP_unsaturatedConductivity.py
from __future__ import print_function
from PSP_readDataFile import readDataFile
import matplotlib.pyplot as plt
import numpy as np
import math

NODATA = -9999

def betacf(a, b, x):
    maxNrIterations = 50
    maxEpsilon = 0.0000003
    am = 1
    bm = 1
    az = 1
    bz = 1 - (a+b) * x / (a+1)
    myEpsilon = 1
    m = 1
    while (myEpsilon > (maxEpsilon * abs(az))):
        if (m > maxNrIterations): return (NODATA)
        d = (m * (b - m) * x) / ((a + 2*m -1) * (a + 2*m))
        ap = az + d * am
        bp = bz + d * bm
        d = -((a + m) * (a + b + m) * x) / ((a + 2*m) * (a + 2*m + 1))
        app = ap + d * az
        bpp = bp + d * bz
        am = ap / bpp
        bm = bp / bpp
        old_az = az
        az = app / bpp
        bz = 1
        m += 1
        myEpsilon = abs(az - old_az)

    return (az)

def incompleteBetaFunction(a, b, x):
    if ((x < 0.) or (x > 1.)):
        return (NODATA)
    if ((x == 0.) or (x == 1.)):
        bt = 0.
    else:
        bt = math.exp(math.lgamma(a + b) - math.lgamma(a) - math.lgamma(b) + a * math.log10(x) + b * math.log10(1. - x))
    if (x < ((a + 1.) / (a + b + 2.))):
        return(bt * betacf(a, b, x) / a)
    else:
        return(1. - bt * betacf(b, a, 1. - x) / b)
```

Calculate the incomplete beta function with the functions of betacf and incompleteBetaFunction

```python

def computeConductivity(currentSe, n, m, Ks):
    p = m + 1. / n
    q = 1. - 1. / n
    z = currentSe ** (1. / m)
    myBeta = incompleteBetaFunction(p, q, z)
    if (myBeta == NODATA):
        return (NODATA)
    else:
        return(Ks * currentSe * (myBeta ** 2.))
```

Compute hydraulic conductivity via Mualem-van Genuchten model

```python

def main():
    A, isFileOk = readDataFile("soil.txt",1,',', True)
    if ((not isFileOk) or (len(A[0]) != 6)):
```

Import the parameters of van-Genuchten model and saturated K for a silt loam soil

```
        print('warning: wrong soil file.')
        return (False)

    VG_alpha = A[0,0]
    VG_n = A[0,1]
    VG_m = A[0,2]
    VG_thetaR = A[0,3]
    thetaS = A[0,4]
    Ks = A[0,5]
```

Define variables for the parameters

```
    A, isFileOk = readDataFile("SWC.txt",1,'\t', False)
    if (not isFileOk):
        print('warning: wrong SWC file in row nr.', A+1)
        return (False)

    waterPotential = A[:,0]
    waterContent = A[:,1]
    conductivity = np.zeros(len(waterContent))
```

Import and define variables for soil water retention curve (theta-psi)

```
    for i in range(0, len(waterContent)):
        currentSe = (waterContent[i] - VG_thetaR) / (thetaS - VG_thetaR)
        conductivity[i] = computeConductivity(currentSe, VG_n, VG_m, Ks)
        if (conductivity[i] == NODATA):
            print ('Error in compute conductivity')
            return (False)
```

Compute the unsaturated hydraulic conductivity from water content and corresponding water potential

```
    plt.figure(figsize=(10,8))
    plt.loglog (waterPotential, conductivity, 'ko')
    plt.xlabel('Water Potential [J kg$^{-1}$]',fontsize=20,labelpad=2)
    plt.ylabel('Hydraulic Conductivity [kg s$^{-1}$
m$^{-2}$]',fontsize=20,labelpad=2)
    plt.tick_params(axis='both', which='major', labelsize=20,pad=6)
    plt.tick_params(axis='both', which='minor', labelsize=20,pad=6)
    plt.show()

main()
```

Plot the unsaturated hydraulic conductivity as a function of water potential as shown in Figure 6.3