Final Project

Edward Ling and Zack He

Objective

The objective is to demonstrate the feasibility of cooling a building using snow storage over the summer.

Design constraints

The snow pile should be at least 50 meters from the building. The snow pile is connected to the building via a heat exchanger adjacent to the building. There are two independent fluid loops. The one from the snow pile comes into the heat exchanger at $2^\circ C$ and leaves at $8^\circ C$. The one from the builing comes at $10^\circ C$ and leaves at $5^\circ C$. The peak cooling capacity (output) should be $1000 kW$. Obviously, the cooling demand fluctuates durung the day. The total heat capacity is assumed to 1000MWh. For heat exchangers a good resource can be found at https://www.engr.mun.ca/~yuri/Courses/MechanicalSystems/HeatExchangers.pdf (https://www.engr.mun.ca/~yuri/Courses/MechanicalSystems/HeatExchangers.pdf) .

The insulation of the pile consists of a thick layer of wood chip mulch covered by a reflective tarp.

You may choose any correlation for the sky temperature (longwave radiation), the simplest one being Hagentoft, 2001:

$$T_{sky,K} = 273.15 + 1.2T_{air,^\circ C} - 14$$

Note that the longwave radiation is mostly if not only outwards (from the pile to the sky). The emissivity of the pile thanks to the reflective blanket is probably small, however make sure that your algorithm does not recreate snow when/if the longwave radiation dominates the total heat rate affecting the snow.

In order to have a somewhat realistic usage of the pile, I made up a cooling function in kW at the bottom of this notebook based on the atmospheric temperature.

Deliverables:

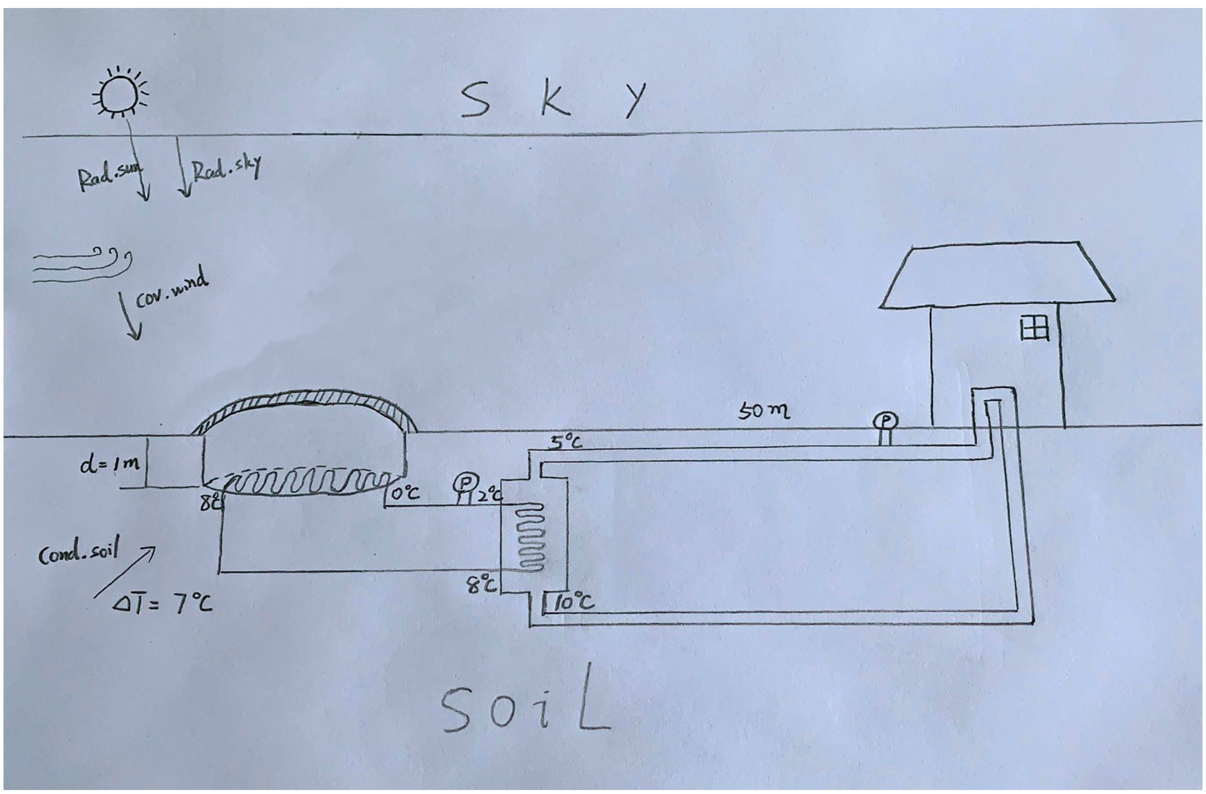
1. Develop a model to demonstrate the feasibility of your concept 40pt
2. Describe your concept in sketches, assumptions and equations 20pt
3. Heat Transfer Analysis of 2 closed Loops 20pt
4. Test your concept against existing weather data 10pt
5. Required Pumping Power 10pt

Assumptions

- Steady state conditions
- 1D heat transfer
- Incompressible flow within pipes
- Uniform soil temperature of $7^{\circ}C$
- Pile has dimensions of $60m \times 60m \times 4m$ for a total volume of $14400m^3$
- All pipe except for pipe with direct contact to snow pile is buried below soil surface
- All pipes are made of CPVC with conductiviy, $k = 0.136 W/mK$
- Heat exchanger is 75% efficient
- Insulation on cold pipe lines
- Reflective blanket has emmisivity of 0.03
- Surface of snow pile is flat
- Distance between the heat exchanger and the building is 45m and between the snow pile and the heat exchanger is 5m

In [1]:
```
from IPython.display import Image
Image(filename='model.png')
```

Out[1]:



Python set-up and useful functions

In [2]:
```
%matplotlib inline


import matplotlib.pyplot as plt
import numpy as np
import math
import scipy.constants as sc
import h5py

import sympy as sym

import SchemDraw as schem
import SchemDraw.elements as e


font = {'family' : 'serif',
        #'color'  : 'black',
        'weight' : 'normal',
        'size'   : 16,
        }
fontlabel = {'family' : 'serif',
        #'color'  : 'black',
        'weight' : 'normal',
        'size'   : 16,
        }

from matplotlib.ticker import FormatStrFormatter
plt.rc('font', **font)

from scipy.constants import convert_temperature
def C2K(T):
    return convert_temperature(T,'Celsius','Kelvin')
def C2F(T):
    return convert_temperature(T,'Celsius','Fahrenheit')
def F2K(T):
    return convert_temperature(T,'Fahrenheit','Kelvin')
def F2C(T):
    return convert_temperature(T,'Fahrenheit','Celsius')
def K2F(T):
    return convert_temperature(T,'Kelvin','Fahrenheit')
def K2C(T):
    return convert_temperature(T,'Kelvin','Celsius')
```

In [3]:
```
import pandas as pd
from pandas import Series

import os
print(os.path.abspath)
summer2017 = pd.read_csv("../wunderground-data/KVTCRAFT2_2017-04-01_2017-10-3
1.csv", delimiter=",",header=0,date_parser=[1])
summer2018 = pd.read_csv("../wunderground-data/KVTCRAFT2_2018-04-01_2018-10-3
1.csv", delimiter=",",header=0,date_parser=[1])
summer2017['time'] = pd.to_timedelta(summer2017['time'].astype(str))
summer2017['date'] = pd.to_datetime(summer2017['date'])
summer2017['date'] = summer2017['date'] + summer2017['time']
summer2017['date'] = pd.to_datetime(summer2017['date'])

# interpolation to 30 mins intervals done for you. You are welcome. You should
still try to understand how it works.
summer2017 = summer2017.set_index('date')
summer2017_tmp = summer2017.resample('30Min').mean()
summer201730mins = summer2017_tmp.interpolate(method='linear')
# weather.set_index('date')
summer201730mins.head(10)

summer2018['time'] = pd.to_timedelta(summer2018['time'].astype(str))
summer2018['date'] = pd.to_datetime(summer2018['date'])
summer2018['date'] = summer2018['date'] + summer2018['time']
summer2018['date'] = pd.to_datetime(summer2018['date'])

# interpolation to 30 mins intervals done for you. You are welcome. You should
still try to understand how it works.
summer2018 = summer2018.set_index('date')
summer2018_tmp = summer2018.resample('30Min').mean()
summer201830mins = summer2018_tmp.interpolate(method='linear')
# weather.set_index('date')
summer201830mins.head(10)
```

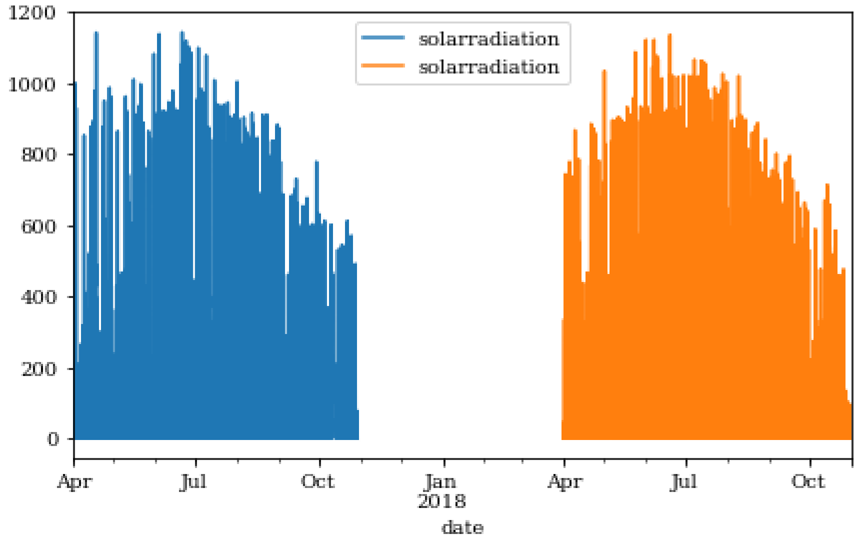
```
<function abspath at 0x000002786955F730>
```

Out[3]:

date	temperature	dewpoint	humidity	wind_speed	wind_gust_speed	wind_dir_degrees	pre
2018-04-01 00:00:00	4.1	-3.9	56.0	0.0	0.0	NaN	1
2018-04-01 00:30:00	3.9	-3.3	59.0	0.0	0.0	NaN	1
2018-04-01 01:00:00	4.1	-4.4	54.0	0.0	0.0	NaN	1
2018-04-01 01:30:00	4.4	-3.3	57.0	0.0	0.0	NaN	1
2018-04-01 02:00:00	4.6	-3.9	56.0	0.0	0.0	NaN	1
2018-04-01 02:30:00	4.4	-2.8	60.0	0.0	0.0	NaN	1
2018-04-01 03:00:00	4.4	-2.8	59.0	0.0	0.0	NaN	1
2018-04-01 03:30:00	4.6	-2.8	60.0	0.0	0.0	NaN	1
2018-04-01 04:00:00	4.6	-2.2	62.0	0.0	0.0	NaN	1
2018-04-01 04:30:00	4.6	-2.2	63.0	0.0	0.0	NaN	1

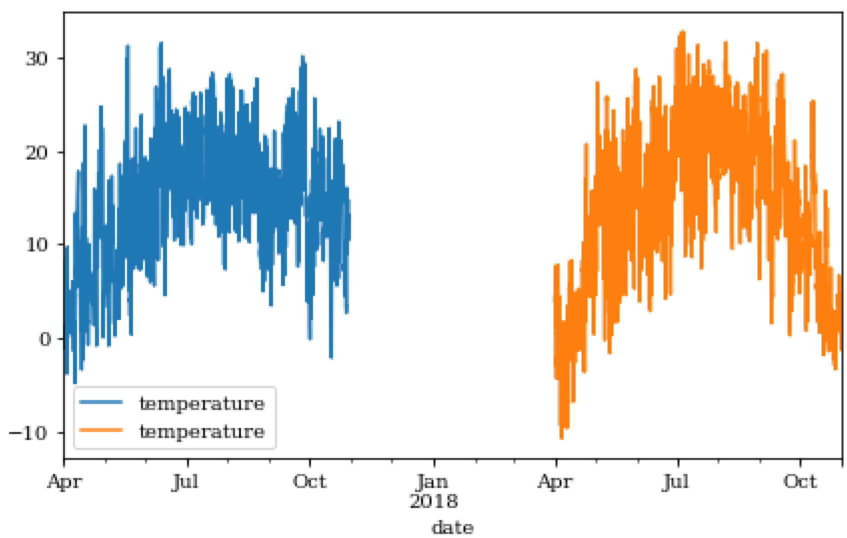
In [4]:
```
ax = summer201730mins.plot( y = 'solarradiation')
summer201830mins.plot(ax = ax, y = 'solarradiation')
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x2786c760e10>



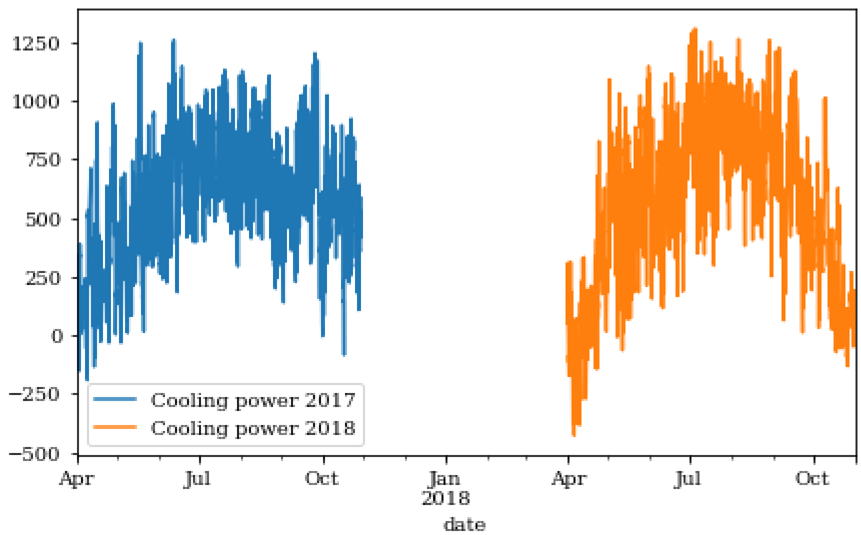
In [5]:
```
ax = summer201730mins.plot( y = 'temperature')
summer201830mins.plot(ax = ax, y = 'temperature')
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x27872471eb8>



In [6]:
```
summer201730mins['Cooling power 2017'] = summer201730mins['temperature']/25*10
00 #function we set, unit in kW
summer201830mins['Cooling power 2018'] = summer201830mins['temperature']/25*10
00 #function we set, unit in kW
ax = summer201730mins.plot(y = 'Cooling power 2017')
summer201830mins.plot(ax = ax, y = 'Cooling power 2018')
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27872889748>
```



Start from the right loop (heat exchanger to buliding loop), the cooling capacity is function of following:

$$\dot{Q}_{cool} = \dot{m}C_p\Delta T$$

therefore, the mass rate of fluid at right loop is:

$$\dot{m} = \frac{\dot{Q}_{cool}}{C_p\Delta T} = \rho A V$$

the velocity of fluid of right loop is

$$V = \frac{\dot{Q}_{cool}}{C_p\Delta T\rho A}$$

By knowing velocity, physical properties of pipe and fluid. The Reynolds number can be found.

If the Reynolds number is larger than 2300. The fluid is turbulent which means that the friction factor can be found.

The head loss for the straight piping should be:

$$h_{loss} = f\frac{L}{D}\frac{V^2}{2g}$$

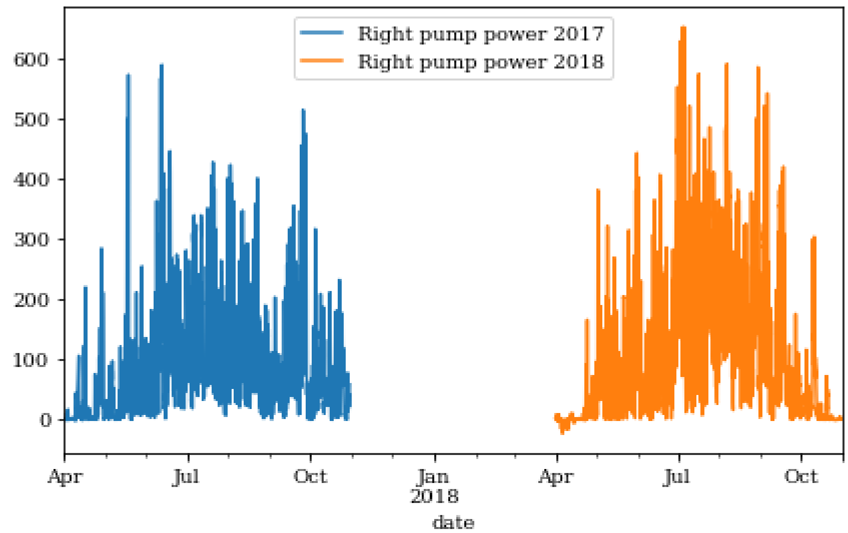
The power required to move the fluid should be:

$$\dot{W} = \dot{m}gh_{loss}$$

In [7]:
```
from NewLibraries import HT_internal_convection as intconv
from NewLibraries import thermodynamics as thermo
import scipy.constants as sccst

Tin_r = C2K(10) #K
Tout_r = C2K(5) #K
Tin_l = C2K(2) #K
Tout_l = C2K(8) #K
Tf_r = (Tin_r+Tout_r)/2
Tf_l = (Tin_l+Tin_r)/2
d_o = 0.2 #m
d_i = 0.18 #m
L_r = 45 #m

# 2017

fluid_r = thermo.Fluid('water',Tf_r)

mdot_r = 1000*summer201730mins['Cooling power 2017']/(fluid_r.Cp*(Tin_r-Tout_r
))
V_r = mdot_r/(fluid_r.rho*math.pi*d_i**2/4)
Um = V_r[:].max()
pipe_r = intconv.PipeFlow(D=d_o, L=L_r, rho=fluid_r.rho, nu=fluid_r.nu, Um=Um)
print("The Reynolds number of right loop is %.2e" %pipe_r.Re)
pipe_r.f_turbulent()

hloss_r = (pipe_r.f*L_r*V_r**2)/(d_i*2*sccst.g) #m
wpump_r = mdot_r*sccst.g*hloss_r
summer201730mins['Right pump power 2017'] = wpump_r #W

print("The total power of pump required on the right loop from April to Novemb
er in 2017 is %.2f kW." %(wpump_r[:].sum()/1000))

# 2018

fluid_r = thermo.Fluid('water',Tf_r)

mdot_r = 1000*summer201830mins['Cooling power 2018']/(fluid_r.Cp*(Tin_r-Tout_r
))
V_r = mdot_r/(fluid_r.rho*math.pi*d_i**2/4)
Um = V_r[:].max()
pipe_r = intconv.PipeFlow(D=d_o, L=L_r, rho=fluid_r.rho, nu=fluid_r.nu, Um=Um)
pipe_r.f_turbulent()

hloss_r = (pipe_r.f*L_r*V_r**2)/(d_i*2*sccst.g) #m
wpump_r = mdot_r*sccst.g*hloss_r
summer201830mins['Right pump power 2018'] = wpump_r #W

print("The total power of pump required on the right loop from April to Novemb
er in 2018 is %.2f kW." %(wpump_r[:].sum()/1000))
ax = summer201730mins.plot(y = 'Right pump power 2017') #unit is W
summer201830mins.plot(ax = ax, y = 'Right pump power 2018')
```

```
The Reynolds number of right loop is 3.35e+05
Pipe wall is assumed to be hydrodynamically smooth
The total power of pump required on the right loop from April to November in
2017 is 884.45 kW.
Pipe wall is assumed to be hydrodynamically smooth
The total power of pump required on the right loop from April to November in
2018 is 1020.67 kW.
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x278749283c8>
```



For the heat exchanger, the efficiency is 75% which means:

$$75\%\dot{m}_L C_{p,water}\Delta T_L - \dot{m}_R C_{p,air}\Delta T_R = 0$$

Therefore, the mass flow rate of water at left loop (Pipe to heat exchanger loop) is:

$$\dot{m}_L = \frac{\dot{m}_R C_{p.air}\Delta T_R}{75\% C_{p.water}\Delta T_L}$$

The velocity of fluid at left loop can be determined:

$$V = \frac{\dot{m}_L}{\rho A}$$

If the Reynolds number is larger than 2300. The fluid is turbulent.

The head loss at 180 degree bend is pretty larger than that in straight pipe. Therefore, we assume that the head loss of straight pipe is neglected. For the 180 degree bend closed loop, the resistance coefficient K is 1.5. Therefore, the head loss for the left loop is:

$$h_{loss_L} = NK\frac{V^2}{2g}$$

where N is the total number of the 180 bend which is 30 in this design.

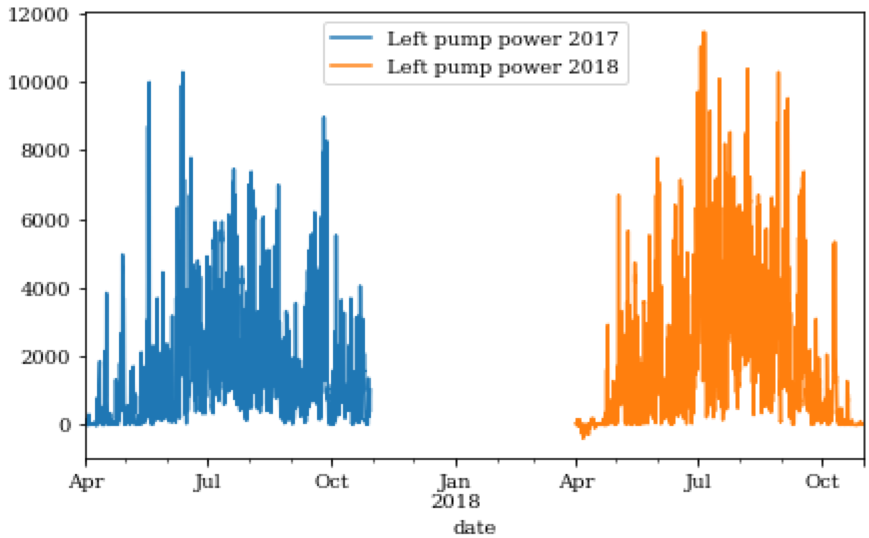
The power required to move the fluid should be:

$$\dot{W} = \dot{m}gh_{loss}$$

In [8]:
```
L_l = 5 #m

fluid_l = thermo.Fluid('water',Tf_l)
mdot_r_17 = 1000*summer201730mins['Cooling power 2017']/(fluid_r.Cp*(Tin_r-Tou
t_r))
mdot_l_17 = -(mdot_r_17*fluid_r.Cp*(Tin_r-Tout_r))/(0.75*fluid_l.Cp*(Tin_l-Tou
t_l))
V_l = mdot_l_17 /(fluid_l.rho*math.pi*d_i**2/4)
Um = V_l[:].max()
pipe_l = intconv.PipeFlow(D=d_o, L=L_l, rho=fluid_l.rho, nu=fluid_l.nu, Um=Um)

print("The Reynolds number of left loop is %1.2e" %pipe_l.Re)
pipe_l.f_turbulent()

N = 30 #number of 180 bends
K = 1.5 #head loss coefficient of 180 degree bend
hloss_l = N*K*V_l**2/(2*sccst.g)
wpump_l = mdot_l_17*sccst.g*hloss_l #W
summer201730mins['Left pump power 2017'] = wpump_l

print("The total pump power required on the left loop from April to November i
n 2017 is %.2f kW." %(wpump_l[:].sum()/1000))

fluid_l = thermo.Fluid('water',Tf_l)
mdot_r_18 = 1000*summer201830mins['Cooling power 2018']/(fluid_r.Cp*(Tin_r-Tou
t_r))
mdot_l_18 = -(mdot_r_18*fluid_r.Cp*(Tin_r-Tout_r))/(0.75*fluid_l.Cp*(Tin_l-Tou
t_l))
V_l = mdot_l_18 /(fluid_l.rho*math.pi*d_i**2/4)
Um = V_l[:].max()
pipe_l = intconv.PipeFlow(D=d_o, L=L_l, rho=fluid_l.rho, nu=fluid_l.nu, Um=Um)

pipe_l.f_turbulent()

N = 30 #number of 180 bends
K = 1.5 #head loss coefficient of 180 degree bend
hloss_l = N*K*V_l**2/(2*sccst.g)
wpump_l = mdot_l_18*sccst.g*hloss_l #W
summer201830mins['Left pump power 2018'] = wpump_l

print("The total pump power required on the left loop from April to November i
n 2018 is %.2f kW." %(wpump_l[:].sum()/1000))
ax = summer201730mins.plot(y = 'Left pump power 2017') #unit is W
summer201830mins.plot(ax = ax, y = 'Left pump power 2018')
```

```
The Reynolds number of left loop is 3.56e+05
Pipe wall is assumed to be hydrodynamically smooth
The total pump power required on the left loop from April to November in 2017
is 15384.12 kW.
Pipe wall is assumed to be hydrodynamically smooth
The total pump power required on the left loop from April to November in 2018
is 17873.04 kW.
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x27876346278>



Heat Fluxes into the Snow Pile

The thermal circuit of pipe from outside pile to heat exchanger at radial direction is:

In [9]:
```python
from NewLibraries import HT_thermal_resistance as res

Rth = []
Rth.append(res.Resistance("$R_{conv,int}$",'W'))
Rth.append(res.Resistance("$R_{cond,pipe}$",'W'))
Rth.append(res.Resistance("$R_{cond,ins}$",'W'))

d = schem.Drawing()

d.add(e.DOT, lftlabel = "$T_m$")
R0 = d.add( e.RES, d='right', label=Rth[0].name )
R1 = d.add( e.RES, d='right', label=Rth[1].name )
R2 = d.add( e.RES, d='right', label=Rth[2].name )
d.add(e.DOT, rgtlabel = "$T_s$")
d.draw()
```

Here the inlet temperature is $1^\circ C$ and outlet is $2^\circ C$. Outside surface temperaure is the temperature of soil which is $7^\circ C$

The required thermal conductivity of the insulation can be found in order to obtain the right insulation material and thickness

$$\frac{T_s - T_{m,o}}{T_s - T_{m,i}} = \exp(-\frac{1}{\dot{m}C_pR_{tot}})$$

where

$$R_{tot} = R_{conv} + R_{cond,pipe} + R_{cond,ins}$$
$$R_{conv} = \frac{1}{hA}$$

where

$$h = \frac{kN_u}{D}$$
$$R_{cond,pipe} = \frac{\ln(r_2/r_1)}{2\pi k_{pipe}L}$$
$$R_{cond,ins} = \frac{\ln(r_3/r_2)}{2\pi k_{ins}L}$$

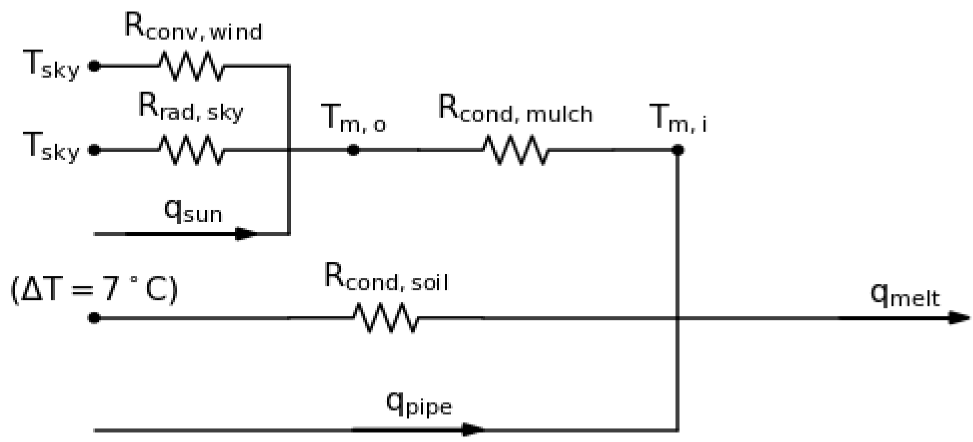
In [10]:
```
T_s = C2K(7)
T_mo = C2K(2)
T_mi = C2K(1)
Ai_l = math.pi*d_i*L_l
Ao_l = math.pi*d_o*L_l
d_ins = 0.22 #m
A_ins = math.pi*d_ins*L_l
k_pipe = 0.19 #W/m2K

pipe_l.laminar_isoflux()
h = fluid_l.k*pipe_l.Nu/d_i
Rconv = 1/(h*Ai_l)
Rcond_pipe = math.log(d_o/d_i)/(2*math.pi*k_pipe*L_l)
Rtot = -1/(math.log((T_s-T_mo)/(T_s-T_mi))*mdot_l_18)
Rcond_ins = Rtot-Rconv-Rcond_pipe

k_ins = math.log(d_ins/d_o)/(Rcond_ins*math.pi*L_l)
print("The conductivity of material used for insulation should be less than %.4f W/mK." %k_ins[:].max())
```

```
The conductivity of material used for insulation should be less than 0.1673 W/mK.
```

Assuming a thickness of 4cm, the thermal conductivity of the insulation must be no greater than $0.17W/mK$. Most fiberglass glass and foam insulation materials have a thermal resistance below 0.035 at regular temperatures.

Thermal circuit for the snow melt:

In [11]:
```
Rth = []
Rth.append(res.Resistance("$R_{conv,wind}$",'W/m'))
Rth.append(res.Resistance("$R_{rad,sky}$",'W/m'))
Rth.append(res.Resistance("$R_{cond,mulch}$",'W/m'))
Rth.append(res.Resistance("$R_{cond,soil}$",'W/m'))
Rth.append(res.Resistance("$R_{cond,pipe}$",'W/m'))

d = schem.Drawing()
d.add(e.DOT, lftlabel = "$T_{sky}$")
R0 = d.add( e.RES, d='right', label=Rth[0].name )
d.add(e.LINE, l = 1.5, d = 'down')
d.push()
R1 = d.add( e.RES, d='left', label=Rth[1].name )
d.add(e.DOT, lftlabel = "$T_{sky}$")
d.pop()
d.push()
d.add(e.LINE, l = 1.5, d = 'down')
L2 = d.add(e.LINE, toplabel = "$q_{sun}$", endpts = [[0, -3], [3, -3]])
d.labelI(L2, arrowofst = 0)
d.pop()
d.add(e.LINE, l = 1, d = 'right')
d.add(e.DOT, label = '$T_{m,o}$')
d.add(e.LINE, l = 1, d = 'right')
R2 = d.add( e.RES, d='right', label=Rth[2].name )
d.add(e.LINE, l = 1, d = 'right')
d.add( e.DOT, label='$T_{m,i}$')
d.add(e.LINE, l = 3, d = 'down')
d.push()
d.add(e.LINE, l = 3, d = 'left')
R3 = d.add( e.RES, d='left', label=Rth[3].name )
d.add(e.LINE, l = 3, d = 'left')
d.add(e.DOT, label = "($\Delta T=7 ^\circ C)$")
d.pop()
d.push()
d.add(e.LINE, l = 2, d = 'down')
d.add(e.LINE, l = 9, d = 'left')
L3 = d.add(e.LINE, toplabel = "$q_{pipe}$", endpts = [[4, -6.5], [6, -6.5]])
d.labelI(L3, arrowofst = 0)
d.pop()
d.add(e.LINE, l = 3, d = 'right')
L2 = d.add(e.LINE, toplabel = "$q_{melt}$", endpts = [[12, -4.5], [13, -4.5]])
d.labelI(L2, arrowofst = 0)
d.draw()
```



Thermal circuit of the pipe underneath the snow pile:

In [12]:
```
Rth = []
Rth.append(res.Resistance("$R_{conv,in}$",'W'))
Rth.append(res.Resistance("$R_{cond,pipe}$",'W'))

d = schem.Drawing()

d.add(e.DOT, lftlabel = "$T_m$")
R0 = d.add( e.RES, d='right', label=Rth[0].name )
R1 = d.add( e.RES, d='right', label=Rth[1].name )
d.add(e.DOT, rgtlabel = "$T_s$")
d.draw()
```

Here the inlet temperature is $8^\circ C$ and outlet is $1^\circ C$. Outside surface temperaure is the temperature of snow which is $0^\circ C$

The log mean temperature is:

$$\Delta T_{\text{lm}} = \frac{T_{m,i} - T_{m,o}}{\ln\left(\frac{T_s - T_{m,o}}{T_s - T_{m,i}}\right)}$$

the total resistance can also be calculated:

$$\frac{T_s - T_{m,o}}{T_s - T_{m,i}} = \exp\left(-\frac{1}{\dot{m} C_p R_{\text{tot}}}\right)$$

The heat from pipe to the snow should be:

$$q = \frac{\Delta T_{\text{lm}}}{R_{\text{tot}}}$$

The heat from solar radiation can be gained from the data provided.

The heat from sky radiation is:

$$q = \epsilon \sigma A (T_{sur}^4 - T_a^4)$$

The heat from wind convection is:

$$q = hA(T_{sur} - T_a)$$

here h is function of wind speed which is:

$$h = 10.45 - V + 10\sqrt{V}$$

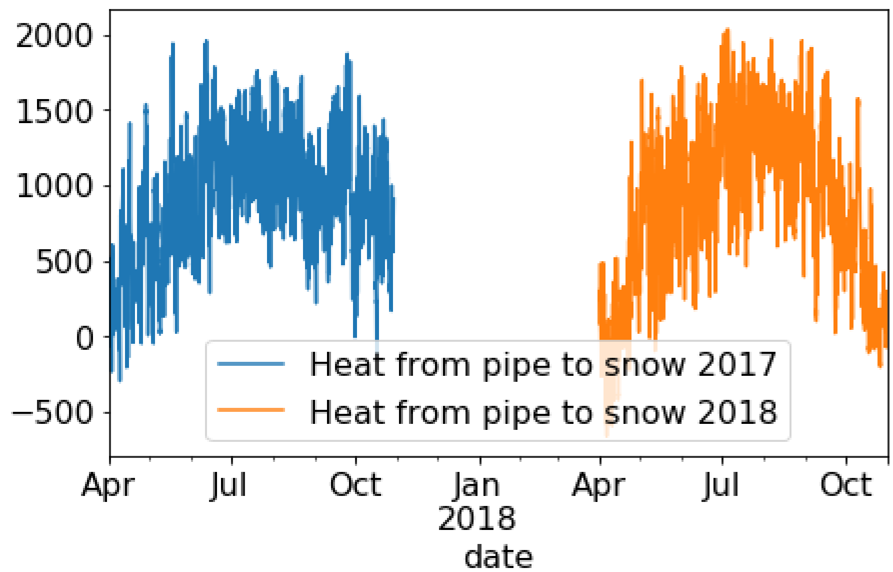
The heat from soil conduction is:

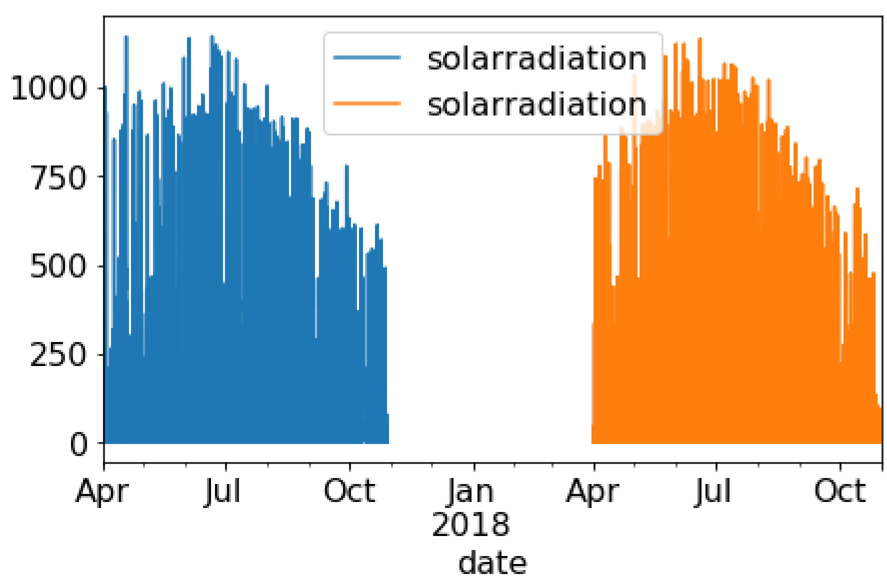
$$q = \frac{k}{d} A \Delta T$$

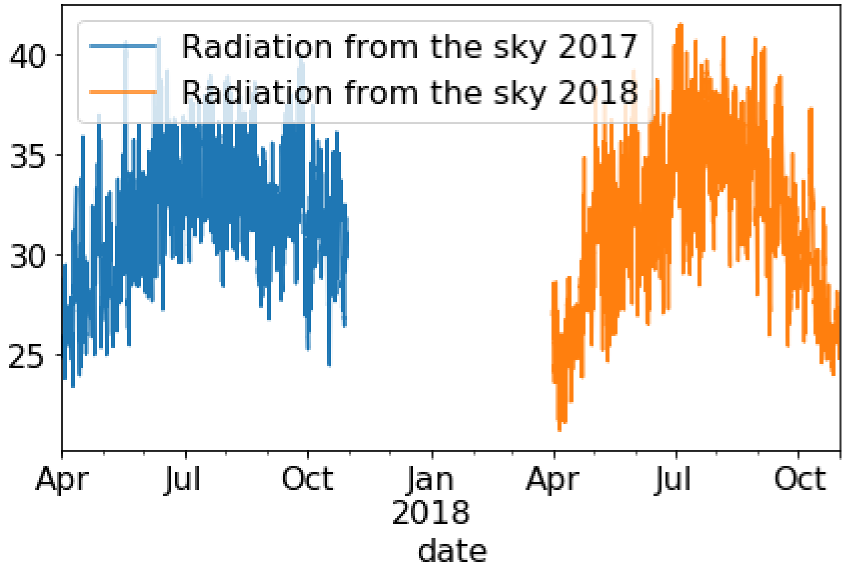
In [17]:
```
# heat from pipe to snow
T_mi = C2K(8)
T_mo = C2K(1)
T_s = C2K(0)
Tlm = (T_mo-T_mi)/math.log((T_s-T_mo)/(T_s-T_mi))
eps = 0.03

Rtot_17 = -1/(math.log((T_s-T_mo)/(T_s-T_mi))*mdot_l_17*fluid_l.Cp)
q_pipe_17 = Tlm/Rtot_17/1000 #kW
Rtot_18 = -1/(math.log((T_s-T_mo)/(T_s-T_mi))*mdot_l_18*fluid_l.Cp)
q_pipe_18 = Tlm/Rtot_18/1000 #kW

summer201730mins['Heat from pipe to snow 2017'] = q_pipe_17
summer201830mins['Heat from pipe to snow 2018'] = q_pipe_18
ax = summer201730mins.plot(y = 'Heat from pipe to snow 2017') #unit is kW
summer201830mins.plot(ax = ax, y = 'Heat from pipe to snow 2018')
print("The mean heat from the pipe in 2017 is %.4f kW." %q_pipe_17[:].mean())
print("The mean heat from the pipe in 2018 is %.4f kW." %q_pipe_18[:].mean())

# solar radiation
ax = summer201730mins.plot( y = 'solarradiation') #unit is kW for the data
summer201830mins.plot(ax = ax, y = 'solarradiation')

# sky radiation
A_plate = 3600 #m2
T_a_17 = C2K(summer201730mins['temperature'])
T_sur_17 = T_a_17*1.2-14
q_sky_17 = eps*sccst.sigma*A_plate*(T_sur_17**4-T_a_17**4)/1000 #kW
summer201730mins['Radiation from the sky 2017'] = q_sky_17

T_a_18 = C2K(summer201830mins['temperature'])
T_sur_18 = T_a_18*1.2-14
q_sky_18 = eps*sccst.sigma*A_plate*(T_sur_18**4-T_a_18**4)/1000 #kW
summer201830mins['Radiation from the sky 2018'] = q_sky_18
ax = summer201730mins.plot(y = 'Radiation from the sky 2017')
summer201830mins.plot(ax = ax, y = 'Radiation from the sky 2018')
print("The mean heat from the sky in 2017 is %.4f kW." %q_sky_17[:].mean())
print("The mean heat from the sky in 2018 is %.4f kW." %q_sky_18[:].mean())

#convection
V_17 = summer201730mins['wind_speed']
h_air_17 = 10.45-V_17+10*V_17**0.5
q_wind_17 = h_air_17*A_plate*(T_sur_17-T_a_17)/1000 #kW
summer201730mins['Convection from wind 2017'] = q_wind_17

V_18 = summer201830mins['wind_gust_speed']
h_air_18 = 10.45-V_18+10*V_18**0.5
q_wind_18 = h_air_18*A_plate*(T_sur_18-T_a_18)/1000 #kW
summer201830mins['Convection from wind 2018'] = q_wind_18

ax = summer201730mins.plot(y = 'Convection from wind 2017')
summer201830mins.plot(ax = ax, y = 'Convection from wind 2018')
print("The mean heat from wind in 2017 is %.2f kW." %q_wind_17[:].mean())
print("The mean heat from wind in 2018 is %.2f kW." %q_wind_18[:].mean())

#conduction of soil
```

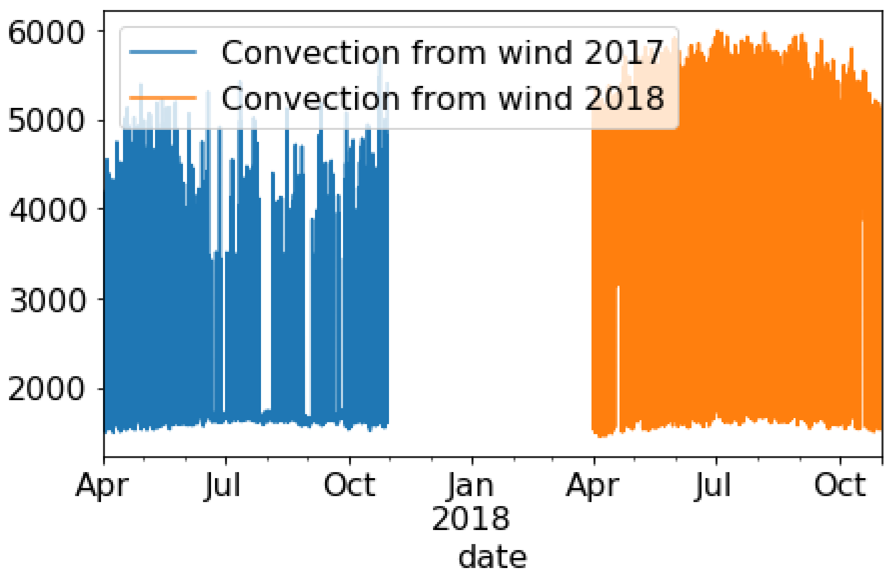
```
DeltaTsoil = 7 #K
ksoil = 1 # W/mk
dsoil = 4 #m
A_soil = 4*60*4+60*60 #m2
q_soil = ksoil/dsoil*A_soil*(DeltaTsoil)/1000
print("The heat from the soil is constant at: %.4f kW." %q_soil)
```

```
The mean heat from the pipe in 2017 is 877.0668 kW.
The mean heat from the pipe in 2018 is 847.4967 kW.
The mean heat from the sky in 2017 is 31.6665 kW.
The mean heat from the sky in 2018 is 31.5169 kW.
The mean heat from wind in 2017 is 2014.88 kW.
The mean heat from wind in 2018 is 4301.58 kW.
The heat from the soil is constant at: 7.9800 kW.
```









In [18]:
```
summer17 = summer201730mins
solarradiation17 = np.array(summer17['solarradiation'])
summer18 = summer201830mins
solarradiation18 = np.array(summer18['solarradiation'])
```

Since we know the velocity height of snow melting is function of heat applied to the snow pile.

$$V_{melt} = \frac{q_{snow}}{\rho_{snow} h_{l,snow}}$$

Where V_{melt} is the velocity of highet

The total volume loss can be calculated.

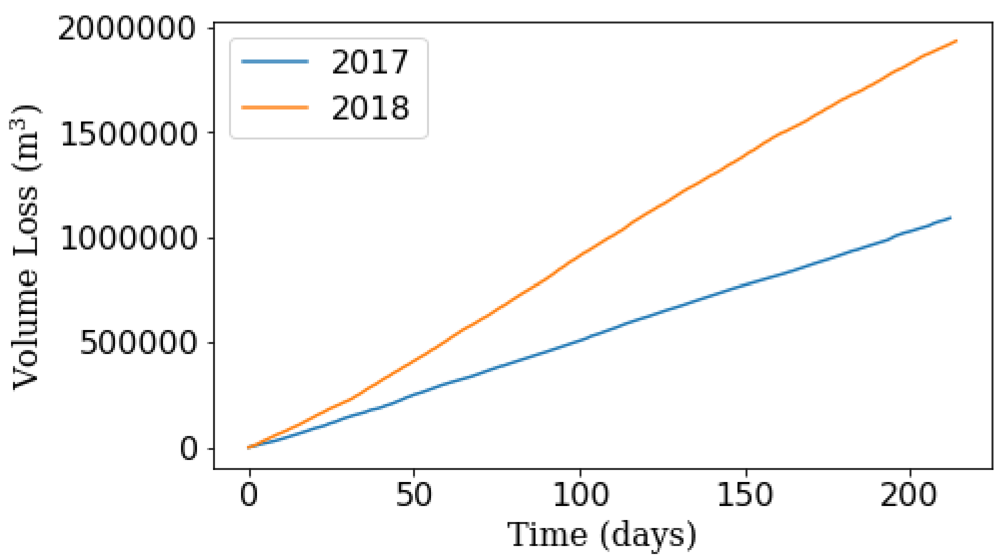
$$Volumeloss = V_{melt} * Time * A$$

therefore,

$$Volumeloss = \frac{Time * A * q_{snow}}{\rho_{snow} h_{l,snow}}$$

In [19]:
```
dt = 1800.0
rho = 550.0 #kg/m^3 range of 500 to 600
h_L= 334000.0 #J/kg-K
A = 3600
def melt(q):
    global dt, rho, h_L
    V = (dt*q*A)/(rho*h_L)
    return V

Vloss17 = melt(solarradiation17+q_sky_17+q_pipe_17+q_wind_17+q_soil)
Vloss18 = melt(solarradiation18+q_sky_18+q_pipe_18+q_wind_18+q_soil)
Vloss17_total = np.cumsum(Vloss17)
Vloss18_total = np.cumsum(Vloss18)
```

In [20]:
```
time17 = np.arange(start = 0,stop=(len(solarradiation17)*1800),step=1800)
time18 = np.arange(start = 0,stop=(len(solarradiation18)*1800),step=1800)
plt.plot((time17/(60*60*24)), Vloss17_total,label='2017')
plt.plot((time18/(60*60*24)), Vloss18_total,label='2018')
plt.xlabel("Time (days)",fontdict = font)
plt.ylabel("Volume Loss ($m^3$)",fontdict = font)
plt.legend()
plt.show()
```



Conclusion

From the model and alanysis we created, the snow pile project is not feasible. We assumed a volume of 14400 cubic meters, this is much less than the smaller total of ~1 million cubic meters of loss. The 2018 volume loss is much higher because wind gust data was used, which increased the convective heat flux from the air. Looking at the various heat fluxes, convection was the greatest.

In order to make this feasible, the most effective thing to do is to shield the snow pile from moving air. This could be done with some large wind diffusers.

One thing that was not included is a back-up air conditioning system that could suppliment the snow pile cooling system. However there were many simplifications made that can potentially increase the heat that goes into the snow pile. Such as the geoetry; for simplicity we assumed a large rectangular prism, however this is not realistic. A further study that uses realistic geometry should be done.