```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        %config InlineBackend.figure_format = 'retina'
```

1. Simple calculations in Python

```
In [25]: print(1+1)
         print(3--3)
         print(1001/100)
         print(1001//10)
         print("Error: cos(0)\nReason: name 'cos' is not defined")
         print(4**4)
         print(np.cos(0))
         print(np.cos(90))
         print(np.cos(np.radians(90)))
         print(4^4)
         print(4**4)
         print(1.0e6)
         print("Error: log(10.0)\nReason: name 'log' is not defined")
         print(np.log(10.0))
         print(np.log10(10.0))
         print(np.pi)
```

```
2
6
10.01
100
Error: cos(0)
Reason: name 'cos' is not defined
256
1.0
-0.4480736161291702
6.123233995736766e-17
0
256
1000000.0
Error: log(10.0)
Reason: name 'log' is not defined
2.302585092994046
1.0
3.141592653589793
```

2. Now, define your own variables

```
In [35]: a=6
         print(a)
         print(10*a)
         print("Error: c = a + b\nReason: b is not defined yet")
         d = 90
         print(a + d)
         print("Error: np.sin(2*D)\nReason: D is case sensitive and isn't the same as
         print(np.sin(2*d))
         print("Error: 2d = 2*d\nReason: 2d isn't a valid variable name")
```

```
6
60
Error: c = a + b
Reason: b is not defined yet
96
Error: np.sin(2*D)
Reason: D is case sensitive and isn't the same as d
-0.8011526357338304
Error: 2d = 2*d
Reason: 2d isn't a valid variable name
```

3. Compute the density

g/cc is grams to cubic centimeter. There are 1000 grams in a kilogram and there are 100,000 cubic centimters in a kilometer.

In [39]:

```
R = 6371 / 1000
M = 5.9736e24 / 100000
p = M / ((4/3)*np.pi*R**3)
p
```

Out[39]:

```
5.514735834031445e+16
```

In [44]:

```
p
```

Out[44]:

```
5.514735834031445e+16
```

In [41]:

```
print(7.8 - p)
print(2.7 - p)
```

```
-5.514735834031444e+16
-5.514735834031445e+16
```

In [ ]:

```
import numpy as np

R = 6371 * 1000
M = 5.9736e24
V = (4/3) * np.pi * R**3
p = M / V
print("Density (kg/m^3):", p)
print("Density (g/cm^3):", p/1000)
```

```
Density (kg/m^3): 5514.735834031448
Density (g/cm^3): 5.514735834031447
```

In [46]:

```
print(7.8 - 5.514735834031447)
print(2.7 - 5.514735834031447)
```

```
2.2852641659685524
-2.8147358340314472
```

Closer to ambient rock density.

4. Vectors in Python

In [59]:
```
a = np.array( [ 1, 2, 3, 3, 5, 5 ] )
print(a)
print(len(a))
print(a.shape)
print(a.shape[0])
print(type(a))

print(2.0*a)
print("Error: a.append(6)\nReason: You can't append an array")
```

```
[1 2 3 3 5 5]
6
(6,)
6
<class 'numpy.ndarray'>
[ 2.  4.  6.  6. 10. 10.]
Error: a.append(6)
Reason: You can't append an array
```

In [65]:
```
list = [ 1, 2, 3, 3, 5, 5 ]
print(list)
print(list[1])
print(len(list))
print(type(list))

list.append(5)
print(list)
list.append('AA')
print(list)
print("Error: 2.0*list\nReason: You can't use a float to multiply; must be i
```

```
[1, 2, 3, 3, 5, 5]
2
6
<class 'list'>
[1, 2, 3, 3, 5, 5, 5]
[1, 2, 3, 3, 5, 5, 5, 'AA']
Error: 2.0*list
Reason: You can't use a float to multiply; must be integer
```

In [66]:
```
tuple = ( 1, 2, 3, 3, 5, 5 )
print(tuple)
print(tuple[1])
print(len(tuple))
print(type(tuple))

print("Error: tuple.append('AA')\nReason: Tuples are immutable")
print(tuple)
print("Error: 2.0*tuple\nReason: You can't use a float to multiply; must be
(1, 2, 3, 3, 5, 5)
2
6
<class 'tuple'>
Error: tuple.append('AA')
Reason: Tuples are immutable
(1, 2, 3, 3, 5, 5)
Error: 2.0*tuple
Reason: You can't use a float to multiply; must be integer
```

In [67]:

```
b = np.zeros(15)
print(b)
c = np.ones(6)
print(c)
print(a+c)
print(a*(c+1))
print(np.sqrt(a))
```

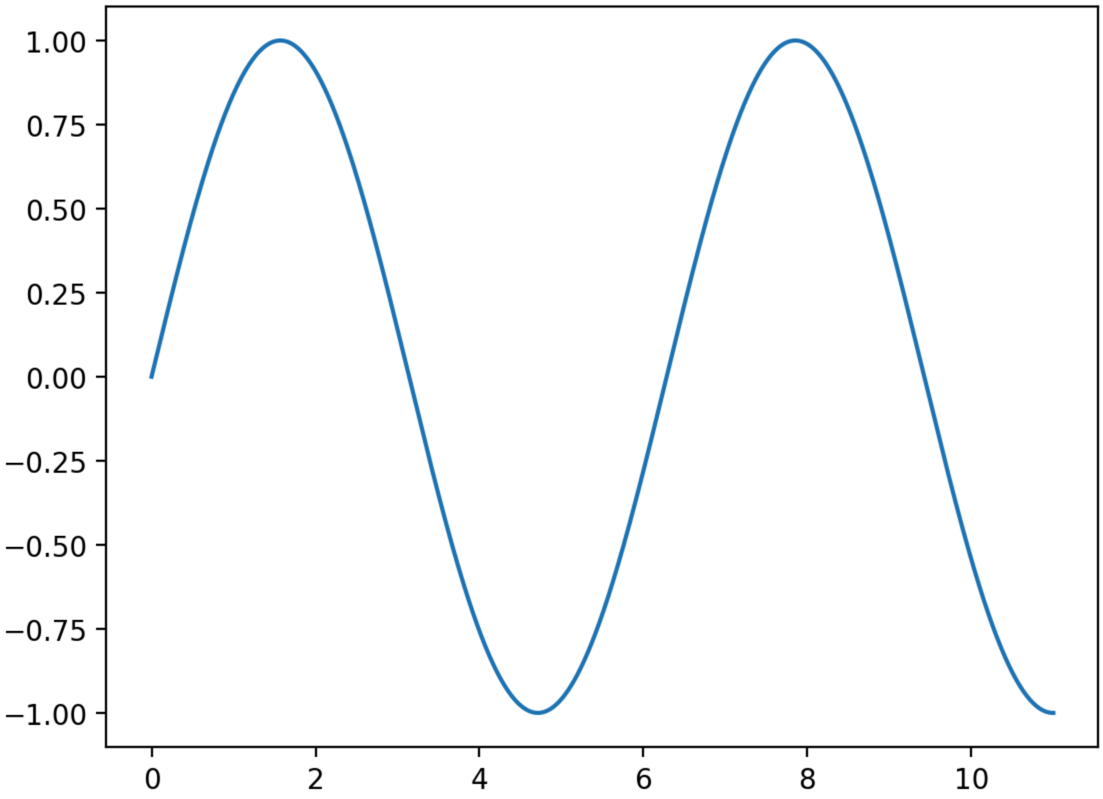
```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[1. 1. 1. 1. 1. 1.]
[2. 3. 4. 4. 6. 6.]
[ 2.  4.  6.  6. 10. 10.]
[1.         1.41421356 1.73205081 1.73205081 2.23606798 2.23606798]
```

5. Generating plots

In [73]:

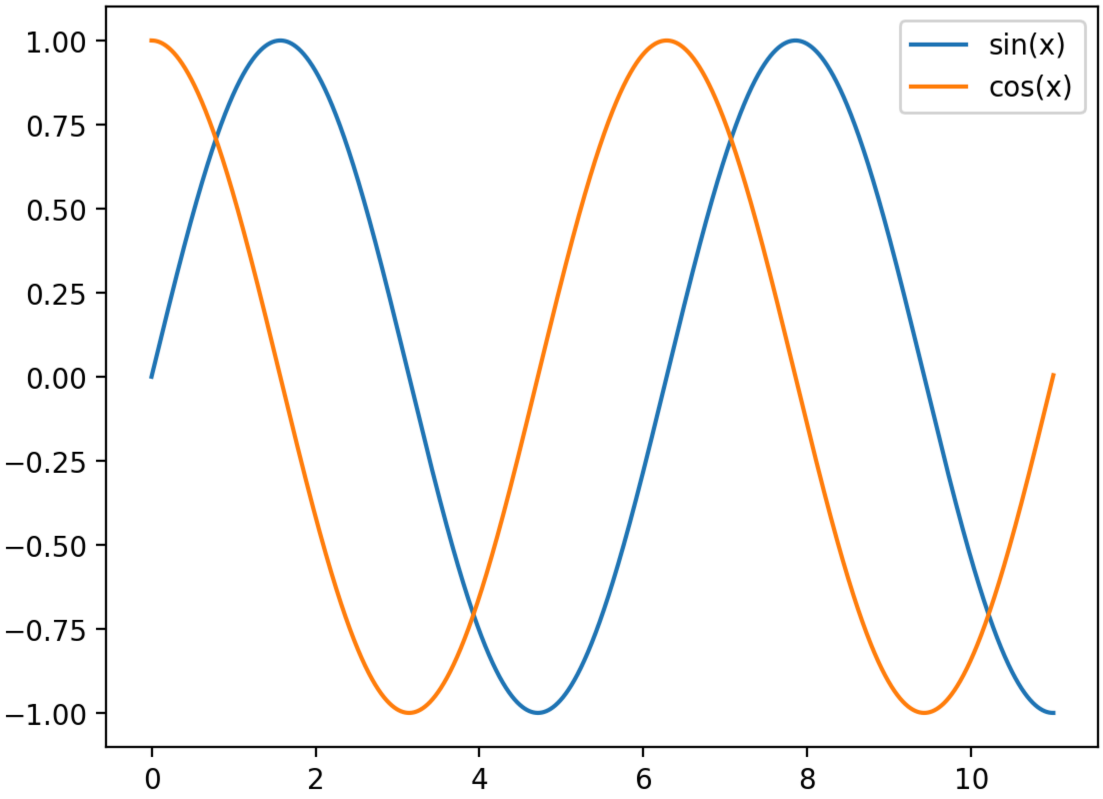
```
# Compute the x and y coordinates for points on a sine curve
x = np.linspace(0.0, 11.0, 10000)
y = np.sin(x)
print(x)
print(y)
# Plot the points using matplotlib
plt.plot(x,y)
plt.show() # In some cases, you need to call plt.show() to make the graph
```

```
[0.00000000e+00 1.10011001e-03 2.20022002e-03 ... 1.09977998e+01
 1.09988999e+01 1.10000000e+01]
[ 0.          0.00110011  0.00220022 ... -0.99999752 -0.99999447
 -0.99999021]
```



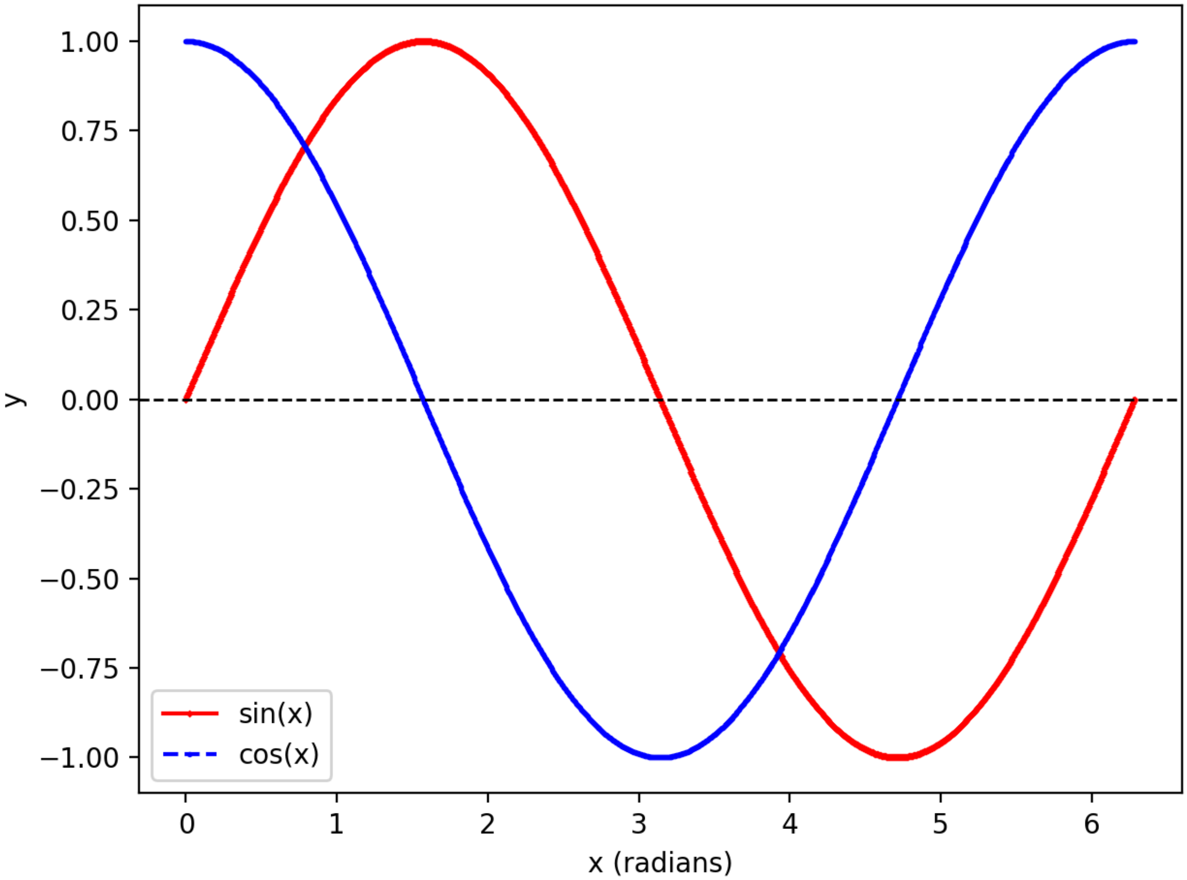
In [74]:

```
y2 = np.cos(x)
# Plot the points using matplotlib
plt.plot(x, y, label='sin(x)')
plt.plot(x, y2,label='cos(x)')
plt.legend()
plt.show()
```



In [83]:

```
import numpy as np
import matplotlib.pyplot as plt

# Use only 10 x values
x = np.linspace(0, 2*np.pi, 1000)
y = np.sin(x)
y2 = np.cos(x)

plt.plot(x, y, 'rD-', linewidth=1.5, markersize=1,
         mec='r', mfc='white', label="sin(x)")

plt.plot(x, y2, 'bo--', linewidth=1.5, markersize=1,
         mec='b', mfc='white', label="cos(x)")

plt.axhline(0, color='k', linestyle='--', linewidth=1)

plt.xlabel('x (radians)')
plt.ylabel('y')
plt.legend()
plt.tight_layout()
plt.show()
```



6. Generate a 3D plot

In [76]:

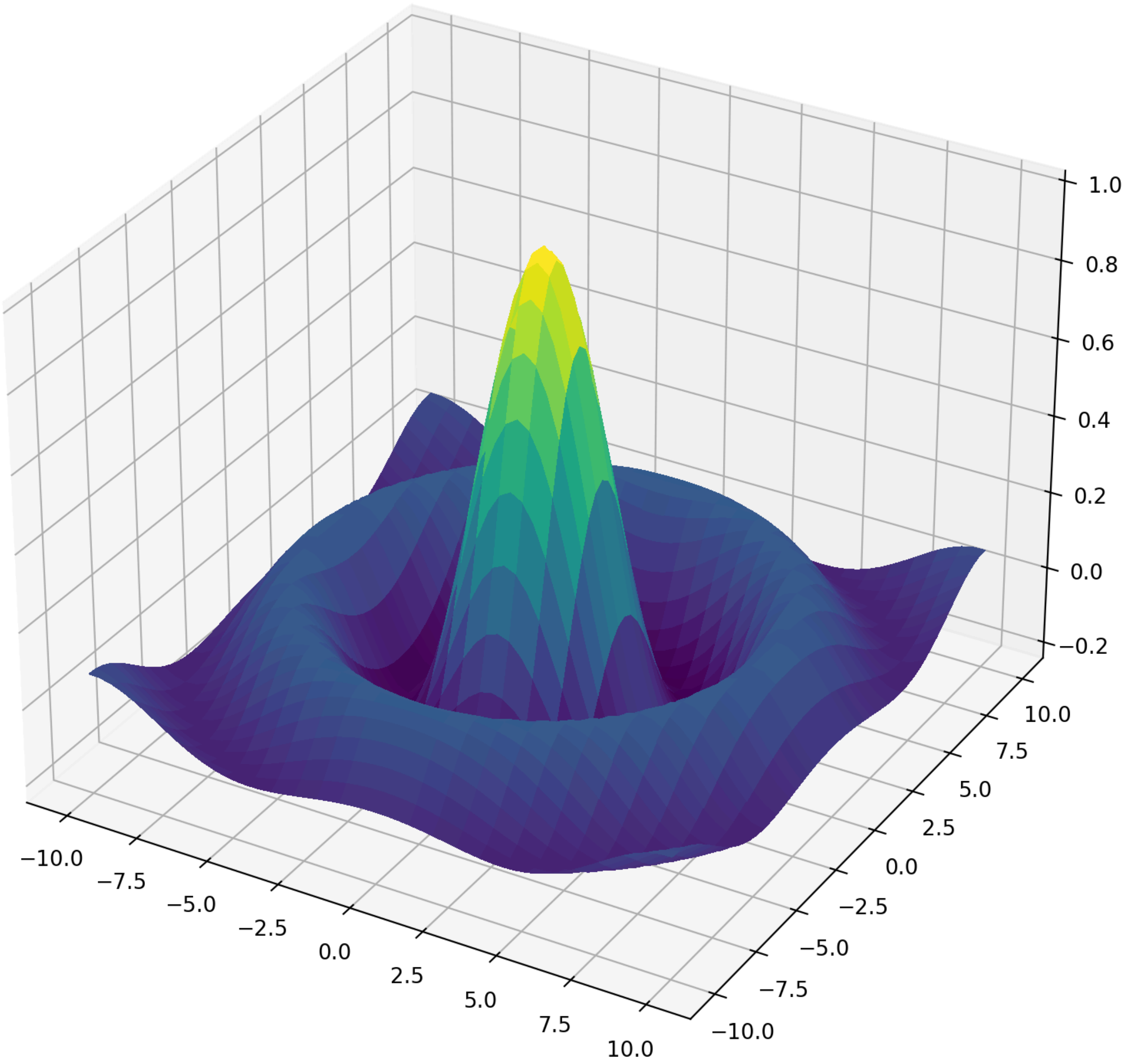
```
n = 51
L = 10.0
x = np.linspace(-L, L, n)
y = np.linspace(-L, L, n)
X, Y = np.meshgrid(x, y)
```

In [77]:

```
R = np.sqrt(X**2+Y**2) + 1e-14 # Why do we need to add 10^-14 ?
Z = np.sin(R) / R
```

In [85]:

```
#the following 3D plot requires some extra libraries
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')
plt.rcParams['figure.figsize'] = [10, 10]
#Try both plot command one after the other
#surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
# linewidth=0, antialiased=False)
surf = ax.plot_surface(X, Y, Z, cmap='viridis',
linewidth=10, antialiased=False,rcount=n)
#resolution is specified by 'rcount' or 'ccount'
plt.show()
```

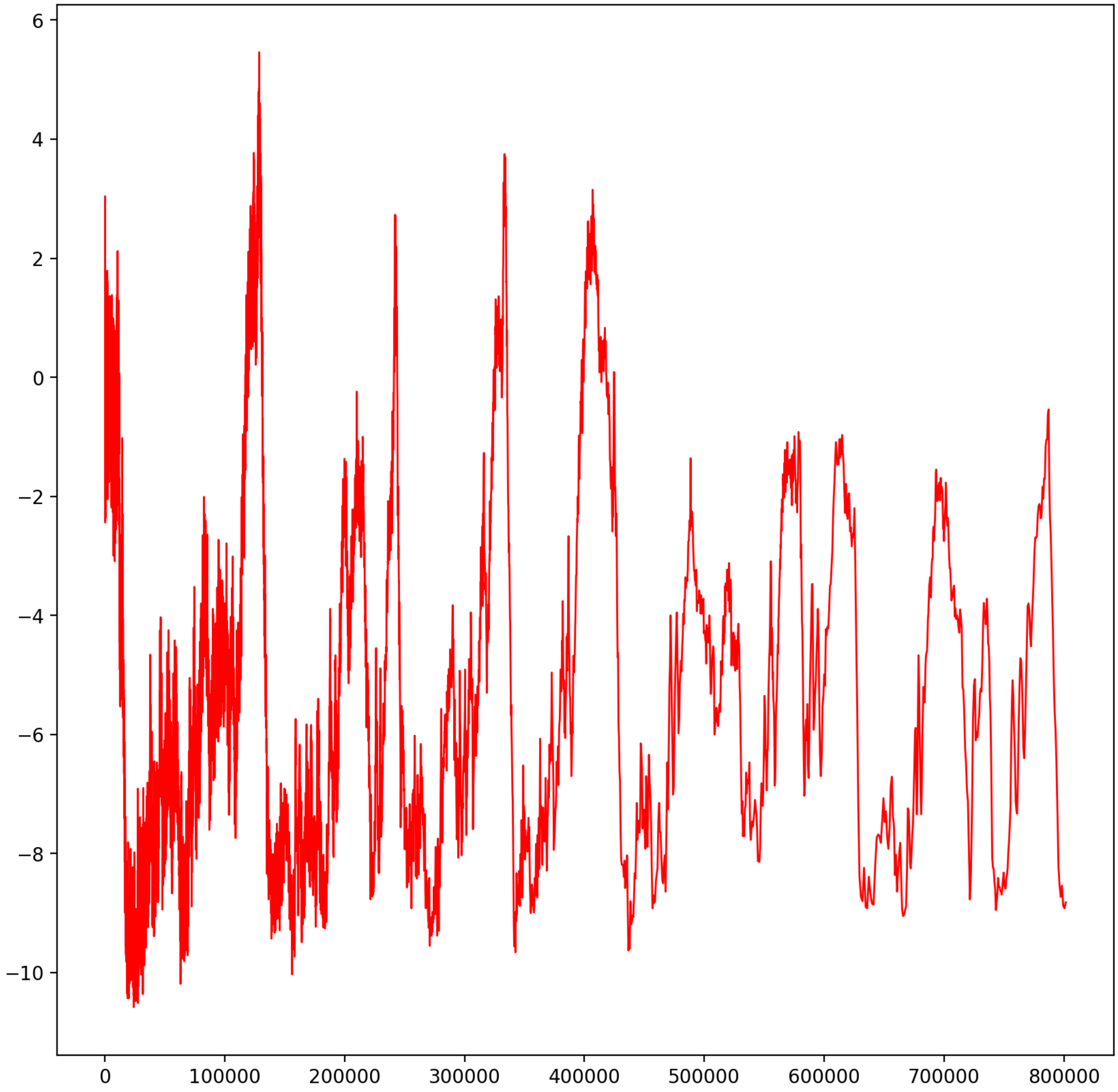


7. Load the data file

In [86]:
```
data = np.loadtxt('ice_core_temperature_data.txt', usecols=(2,4))
#why 2 and 4 instead of 3 and 5?
print(data)
```

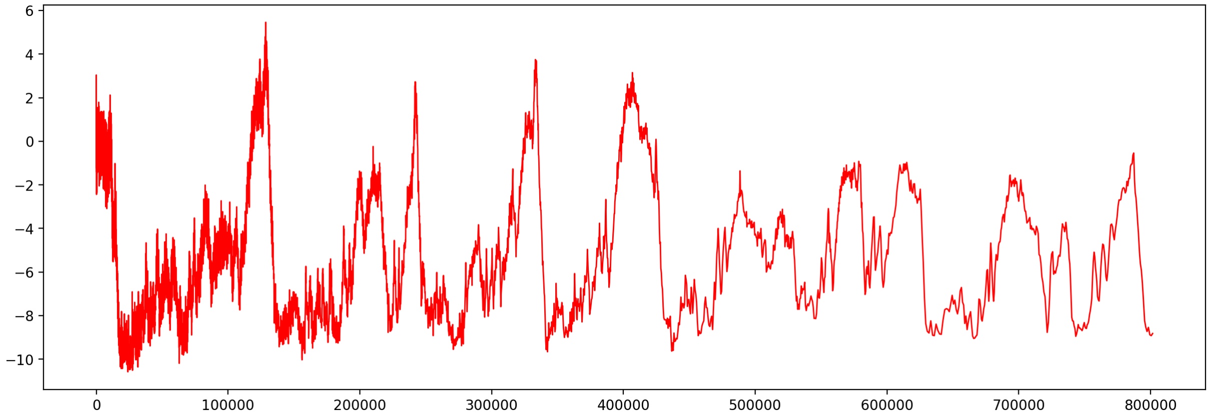
```
[[ 3.837379e+01  8.800000e-01]
 [ 4.681203e+01  1.840000e+00]
 [ 5.505624e+01  3.040000e+00]
 ...
 [ 7.995010e+05 -8.880000e+00]
 [ 8.005890e+05 -8.920000e+00]
 [ 8.016620e+05 -8.820000e+00]]
```

In [87]:
```
plt.plot(data[:,0],data[:,1],'r-',linewidth=1,markersize=0,
mec='r',mew=2, mfc='pink',label="sin(x)");
plt.show()
```



In [88]:

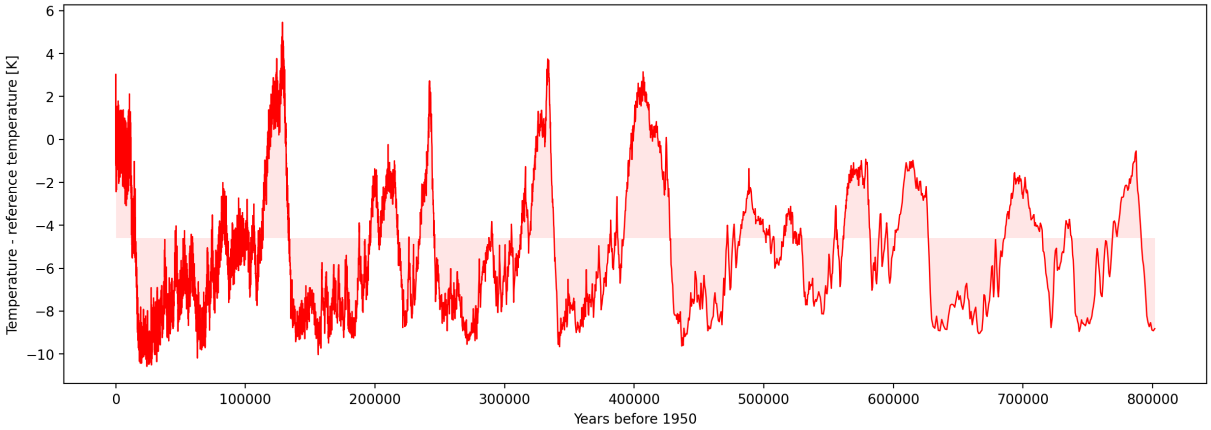
```
#Hey, this plot is too small. I cannot see anything!
plt.rcParams['figure.figsize'] = [15, 5]
plt.plot(data[:,0],data[:,1],'r-',linewidth=1,markersize=0,
mec='r',mew=2, mfc='pink',label="sin(x)");
plt.show()
```



In [89]:
```
temp = data[:,1]
print(sum(temp))
print(temp.shape)
average = sum(temp) / temp.shape[0]
print(average)
```

```
-26496.61999999997
(5785,)
-4.580228176318059
```

In [90]:
```
plt.rcParams['figure.figsize'] = [15, 5]
plt.plot(data[:,0],data[:,1],'r-',linewidth=1,
markersize=0,mec='r',mew=2, mfc='pink',label="sin(x)");
plt.fill_between(data[:,0],data[:,1],
average,color=(1.0,0.90,0.90),lw=0)
plt.xlabel('Years before 1950')
plt.ylabel(r'Temperature - reference temperature [K]')
plt.show()
```


Temperature - reference temperature [K]
Years before 1950