数値計算の基礎と地球惑星科学への応用

II. プレートの冷却の数値シミュレーション・その２

プレートの熱的進化を今度は Python プログラムによってシミュレートしてみよう．このプログラムでは，熱伝導方程式の解をフーリエ級数によって表現している．

10. Python を利用するための準備

Python を利用するために，パソコンに Python ディストリビューションの 1 つ Anaconda を導入する。

10.1 Python

Python は高級プログラミング言語の 1 つで，文法が簡単なためにプログラミングの学習に向いた言語である。人工知能の研究に利用されており，人工知能に関する多くのライブラリが開発されている。このためプログラミング入門用言語として人気が高い。Python は次のような特徴を持つ。

(1) 高級プログラミング言語

(2) 開発者はグイド・ヴァンロッサム

(3) 文法を極力単純化してコードの可読性が高く書きやすい

(4) 標準ライブラリやサードパーティ製のライブラリ、関数など、さまざまな領域に特化した豊富で大規模なツール群

(5) オブジェクト指向、命令型、手続き型、関数型などの形式で書ける

(6) Web アプリケーションやデスクトップアプリケーションなどの開発はもとより，システム用の記述 (script) や、各種の自動処理、理工学や統計・解析などに応用

(7) Python のリファレンス実装である CPython は、フリーかつオープンソースのソフトウェア

(8) 現在，Python 2.7 と 3.7 の 2 つの系統

10.2 Anaconda

Python には開発ツールなどを含む様々な実装があり，それが有料あるいは無償で配布されている。それぞれの実装をディストリビューションと呼ぶ。Anaconda は Python ディストリビューションで最もよく使われているものである。Jupyter Notebook という開発ツールが実装されていて，グラフなどの出力もできる。

10.3 Anaconda のインストール

早速，Anaconda をインストールしてみよう。インストールの前に，Windows の場合，利用しているアカウントがアルファベット(日本語でない)ことを確認する。日本語のアカウントの場合は，英語のアカウントを用意する。

(1) 次の URL にアクセスする。

https://www.anaconda.com/products/individual

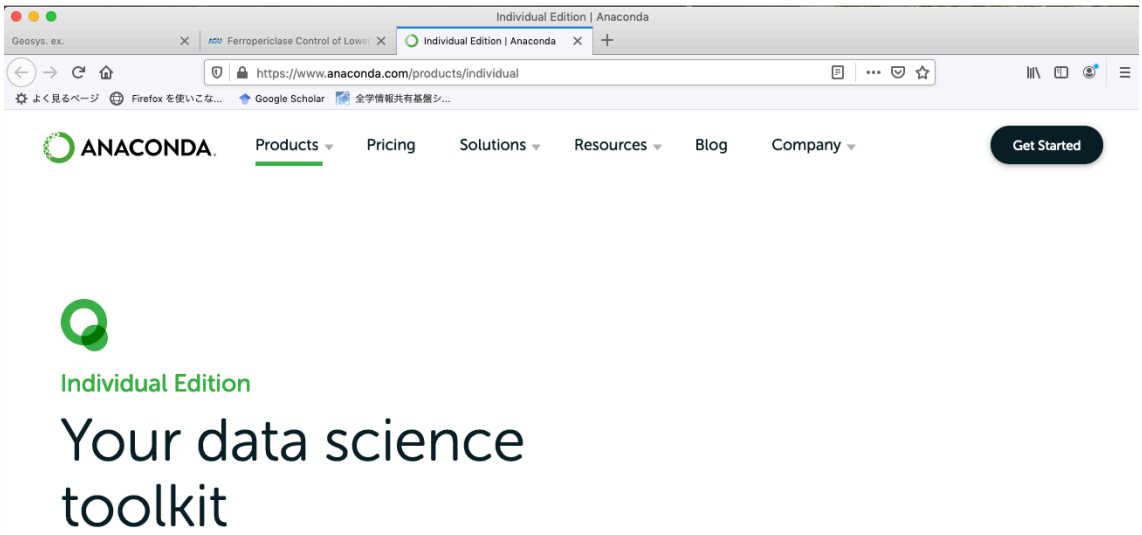

図 10.1 Anaconda 公式ダウンロードサイト

(2) 下の方へスクロールして OS にあったインストーラを選ぶ。

図 10.2 様々な OS に対応したインストーラ

(3) ダウンロードしたインストーラを起動する

図 10.3 Anaconda インストーラが起動

(4) デフォルトインストールで良いので[Next]を押し続ける

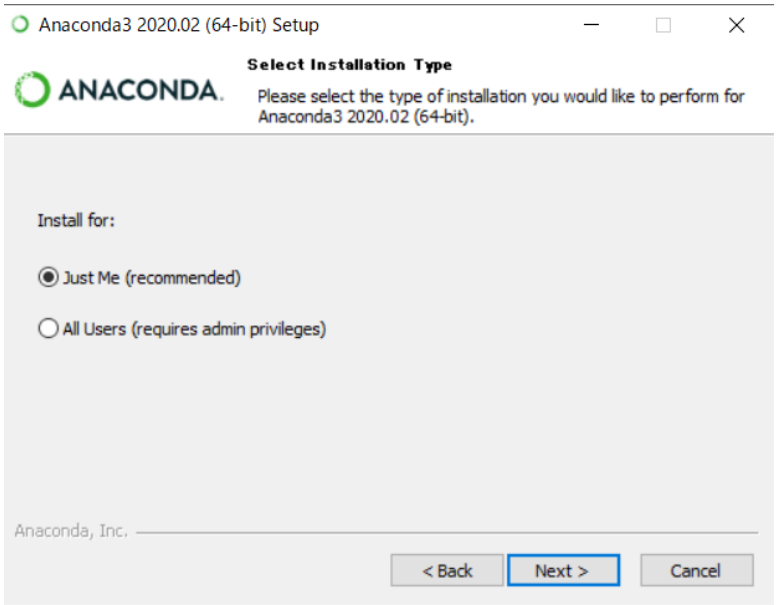

図 10.4 Anaconda インストーラが起動

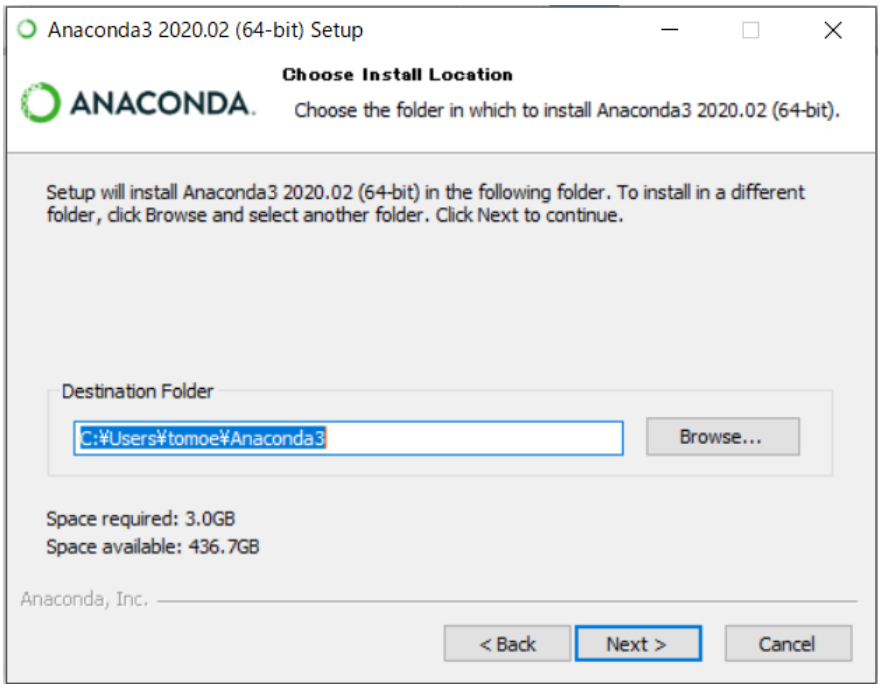

図 10.5 Anaconda のインストール先選択・デフォルトで良い

(4) インストーラを終了する。

(5) ドキュメントフォルダの中の“Python Scrips”を確認。

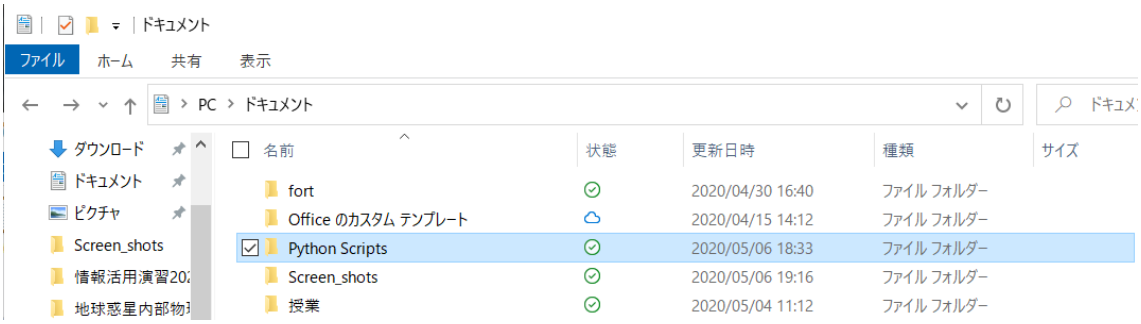

図 10.6 Anaconda のインストール先選択

(6) スタートメニューに Anaconda フォルダとその中のアプリケーションはできていることを確認。

10.3 インストール後のテスト

(1) [Anaconda3 (64bit)]→[Anaconda Powershell Prompt (Anaconda3)]をクリック
　Powershell Prompt では Unix コマンドが使える
(2) Python インタプリタが使えるコマンドプロンプトが起動する
(3) “python –version”と入力して python のバージョンを確認する

図 4.7 Anaconda Pronpt の起動と Python のバージョンを確認

11. Python インタプリタ

11.1 Python インタプリタの起動とプログラムの実行

(1) “python”と入力すると python インタープリタが起動する
(2) プロンプトが“>>>”に変わる
(3) 次のように入力する。1 行毎にエンターキーを押す

```
>>> a = 3
>>> b = 1
>>> c = a + b
>>> print ( c )
```

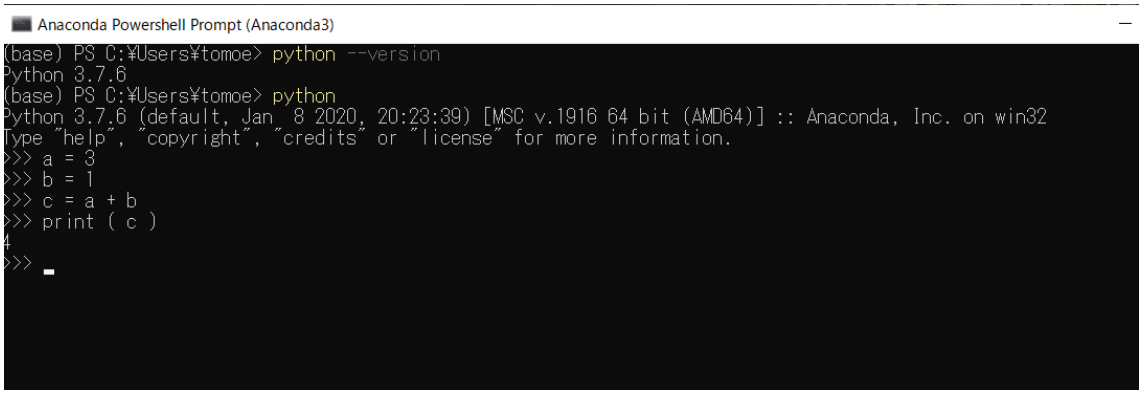

図 11.1 Python インタプリタとプログラムの逐次入力・実行

(4) 上のように計算結果が表示される

(5) [Ctrl]+[D]を押すと python インタプリタが終了する

11.1 プログラムの保存と保存したプログラムの実行

(1) Explorer（ファイルブラウザ）で Python プログラムを保存する場所を作る

[右クリック]→[新規作成]→[フォルダー]

例えば，`C:Users¥`*`yourID`*`¥Documents¥python_prog` など

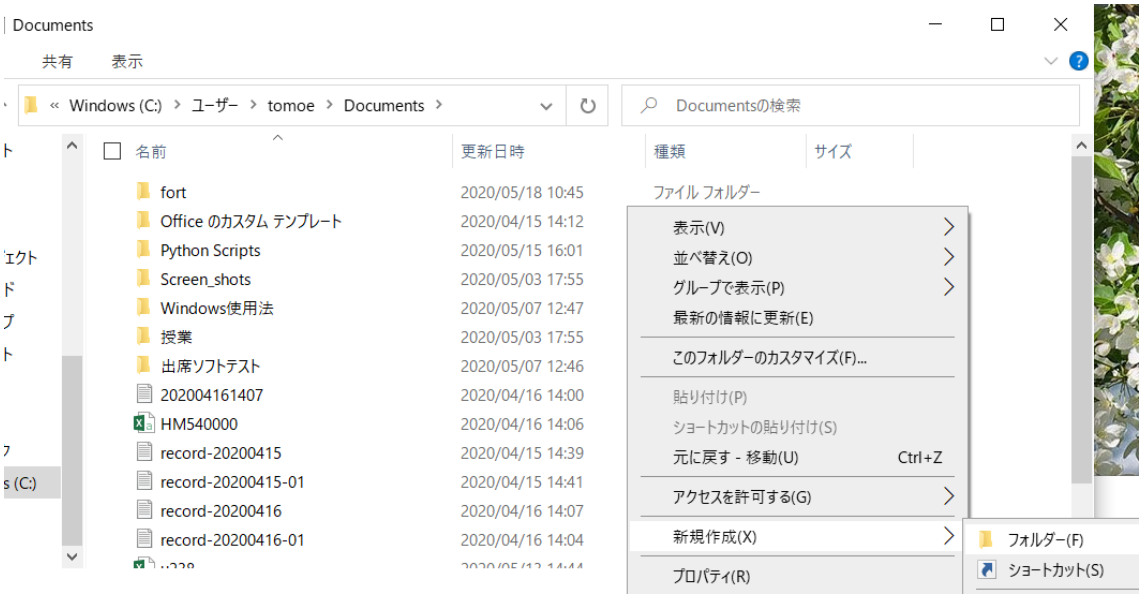

図 11.2 Python プログラム保存先の作成

(1) 「メモ帳」や「TerraPad」などのテキストエディタを起動する

「メモ帳」は[スタートメニュー]→[Windows アクセサリ]の中にある

(2) 「メモ帳」などが起動したら，**5.1** のプログラムを入力する。

(3) [名前をつけて保存]を選ぶ

(4) プログラム保存フォルダ `C:Users¥`*yourID*`¥Documents¥python_prog` に移動する

(4) ファイルの種類を選ぶプルダウンメニューで，[すべてのファイル]を選ぶ

(5) ファイルの名前を入力，[保存]ボタンを押す

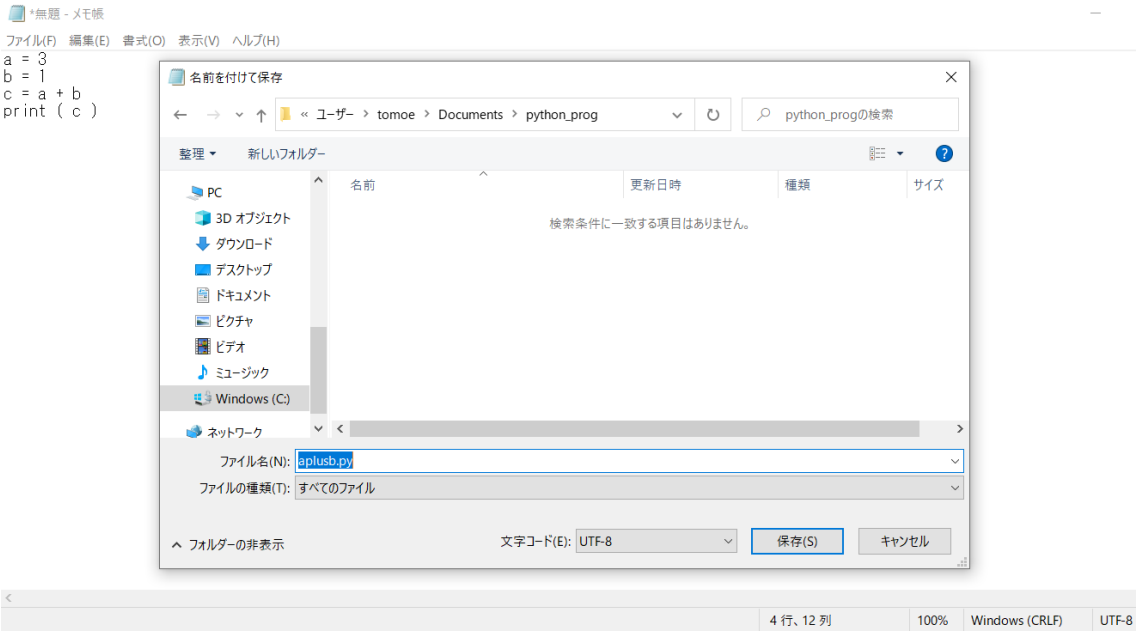

図 11.3 Python プログラム保存先の作成

(6) 先ほど python インタプリタを動かした Anaconda PowerShell Prompt を前面に持ってくる

(7) プロンプトに現在のフォルダがでているので，

`¥Users¥`*yourID*

になっているか確認する

(8) 次のコマンドでフォルダを移動

```
> cd Documents¥python_prog
```

(9) `ls` または `dir` と入力して先ほど入力したプログラムがこの場所に保存されているか確認

(10) 次のように python 実行コマンドを入力して，プログラムを実行する

```
> python aplusb.py
```

(11) 結果が表示されているか確認する

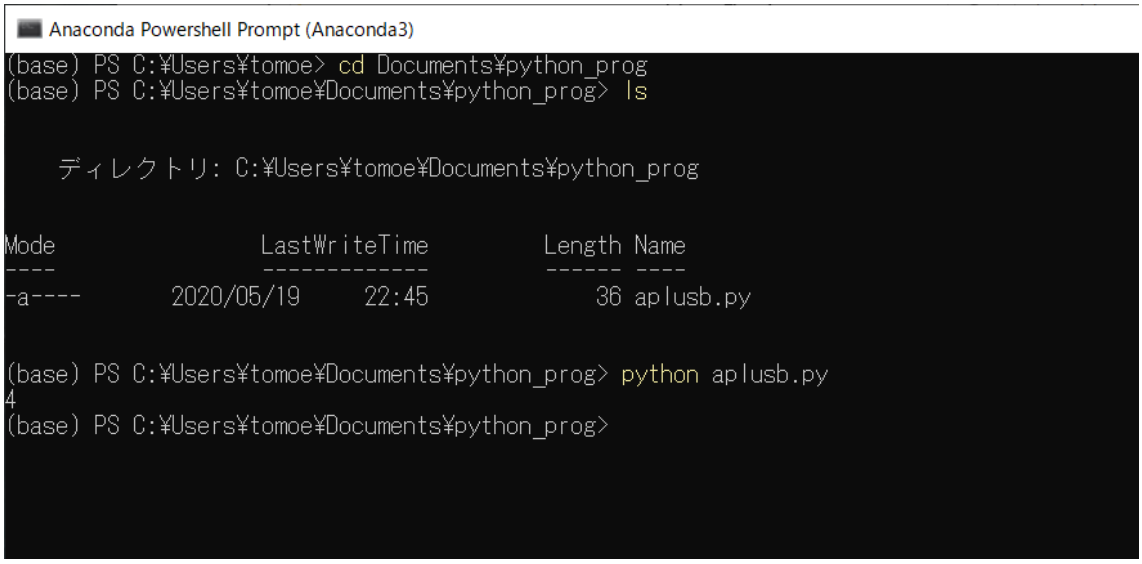

図 11.4 プログラム確認と実行

12. Jupyter Notebook 使用法

12.1 Jupyter Notebook の基本

(1) [スタートメニュー]→[Anaconda3]]→[Jupyter Notebook]を選ぶ

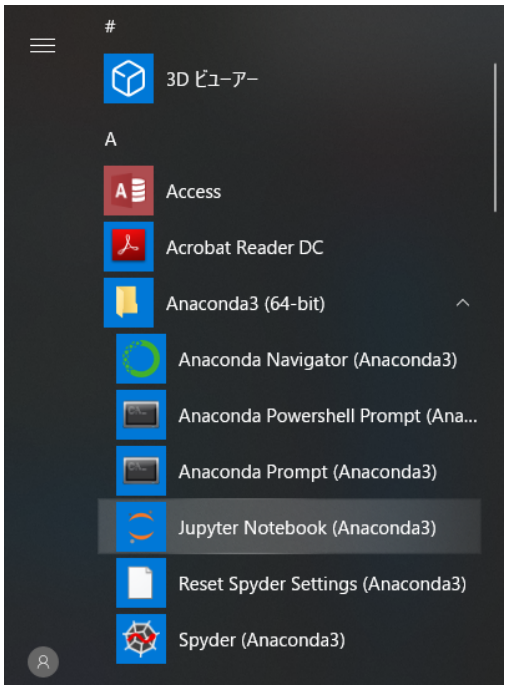

図 12.1 Anaconda インストール後のスタートメニュー

(2) Jupyter Notebook がブラウザ上で起動する。起動すると，Jupyter Notebook のファイルブラウザが表示される。

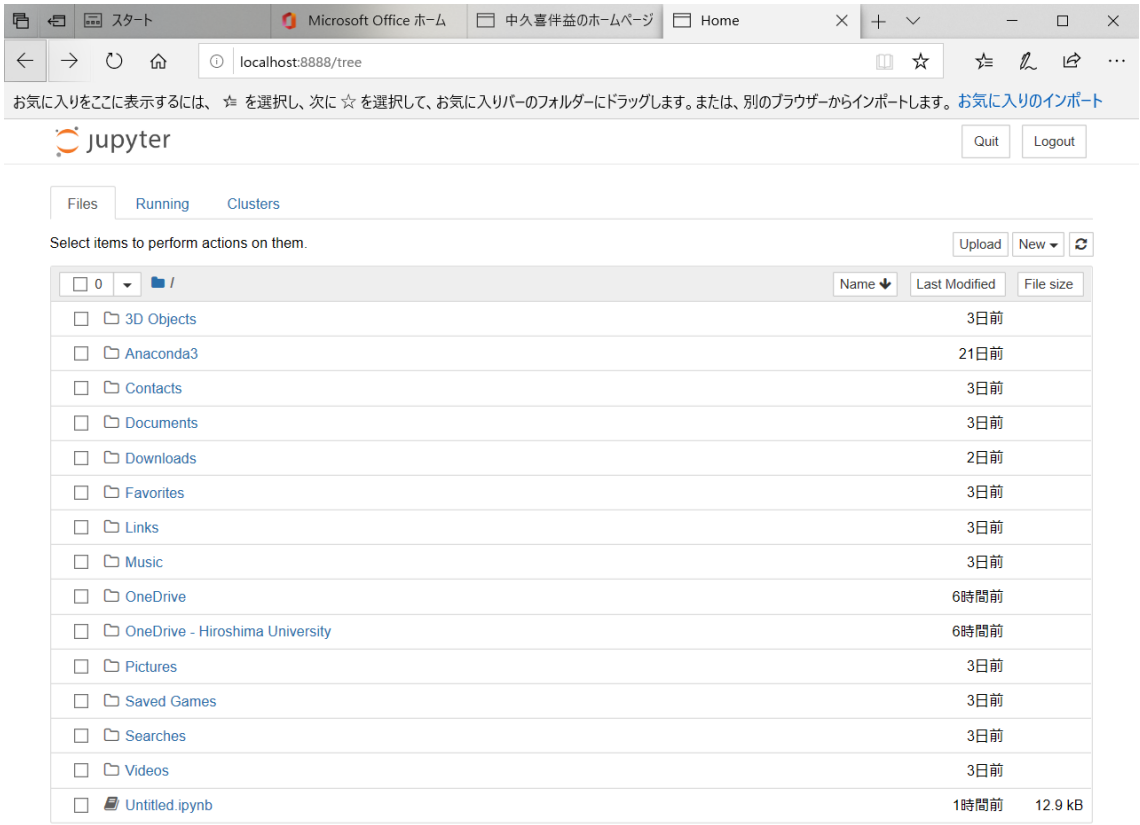

図 12.2 Jupyter Notebook が起動したブラウザ (Microsoft Edge)

(3) 右の[New]から[Python 3]を選ぶ

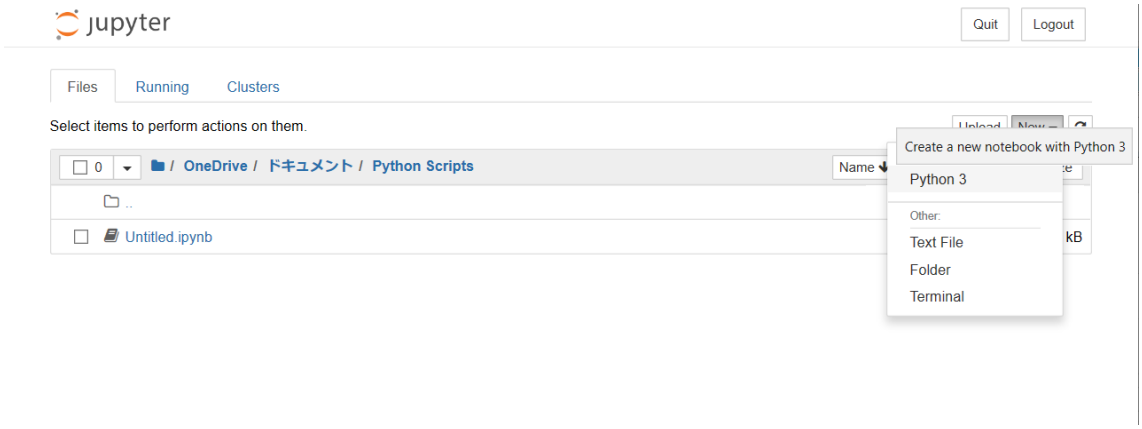

図 12.3 新しいプログラムの選択

(4) ブラウザ画面上にある横長の四角い枠の中に，1 行ずつプログラムを入力する。

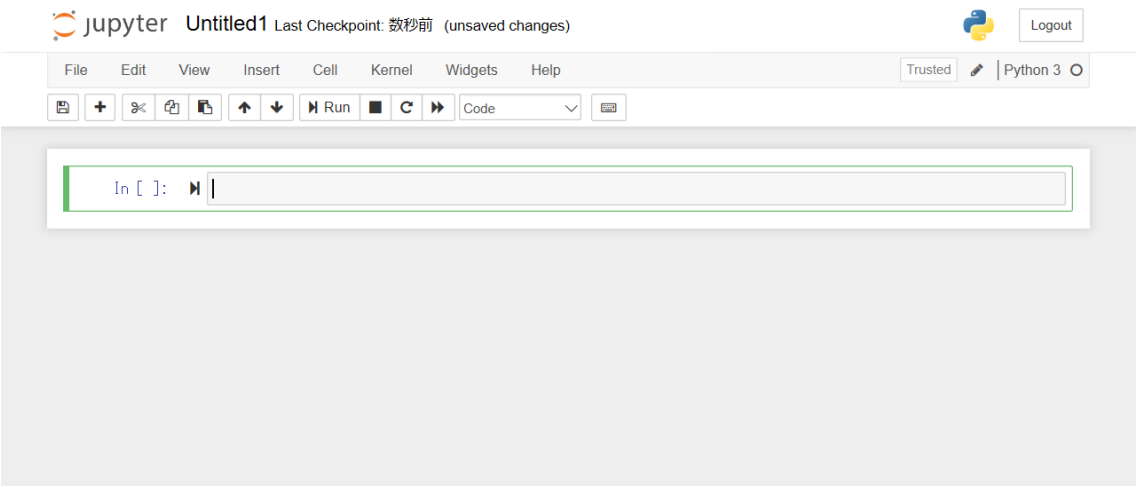

図 12.4 Jupyter Notebook が起動したブラウザ (Microsoft Edge)

(5) [Enter]キーを押すと，行が自動的に増えていくので，次の行が入力できる。

In [5]:
```
%matplotlib inline
import matplotlib.pyplot as plt

# データ
x = list( range( 0, 5 ) )
y = [ ]
for i in range( 5 ):
    y.append( 2 * x[ i ] ** 2 )    # y = 3x -24

# グラフ
plt.plot( x, y )
plt.grid( color = '0.8' )
plt.show
```

図 12.5 プログラム入力中の表示

(6) 入力が終わったら，[Run]ボタンを押す。正しく入力されているときにはプログラムが実行される。このプログラムではグラフが表示される。

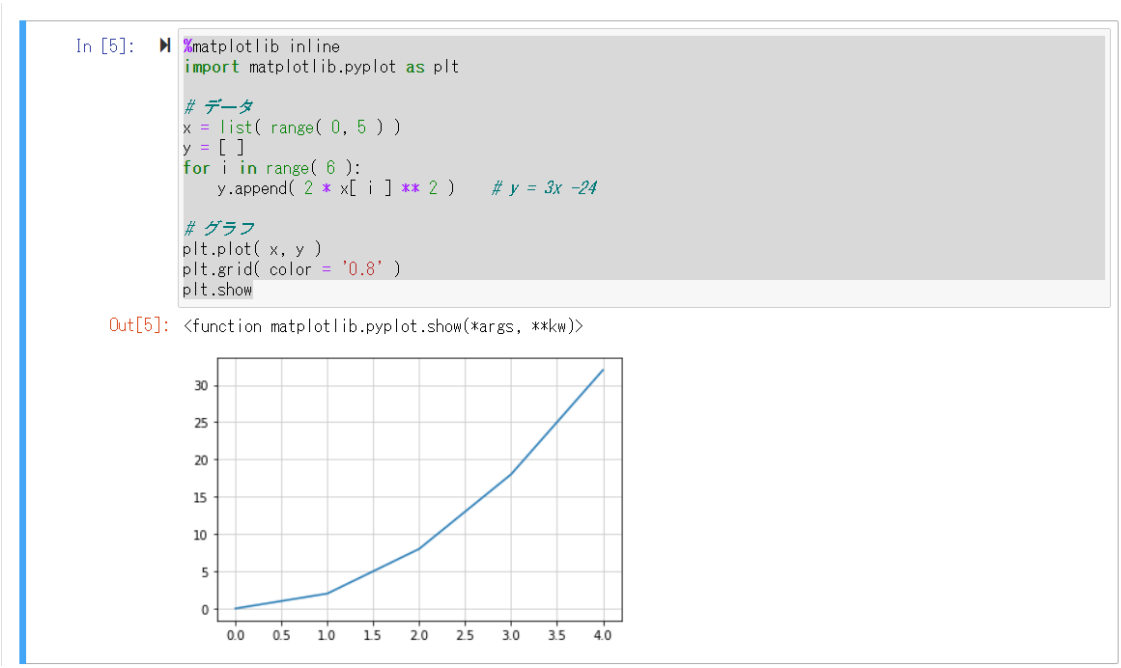

図 12.6 実行後の表示

(5) ファイルとして保存するときは“Untitled”をクリックするか，[File]メニューから[Rename]を選ぶ (名前をつけて保存ではない)。

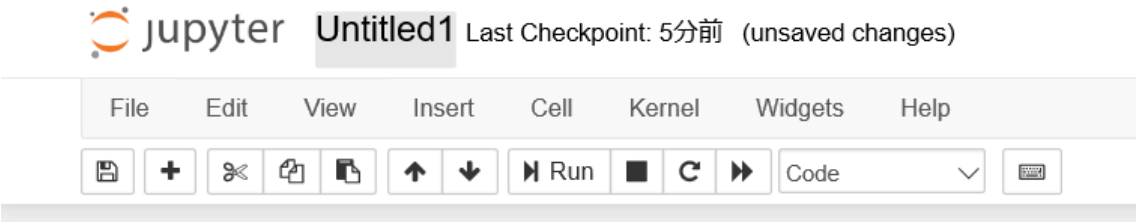

図 12.6 ファイルの保存

(6) ここでは，“sqfun_1 という名で保存した。 ここでは，python のファイル(テキストファイル)でなく，Jupyter Notebook のファイルとして保存する（このため，ファイル名に python 拡張子.py つけてはいけない。自動的に.ipynb になる)。

図 12.7 ファイル保存後の表示

(7) 再度ファイルを開くときには，起動画面のファイル表示のところからファイルを選ぶ（クリックする)。

図 12.8 Jupyter Notebook ファイルブラウザの表示

13. Python によるプレートの冷却の数値シミュレーション

Python プログラムを動かしてプレート冷却のシミュレーションを行う．

13.1 Python プログラムによる数値計算

ここでは，フーリエ級数による熱伝導方程式の解を離散化した式

$$T_i(t) = T_0 + (T_M - T_0)\left\{\frac{z_i}{L} + \sum_{n=1}^{M}\frac{2}{n\pi}\sin\left[\frac{n\pi z_i}{L}\right]\exp\left[\frac{n^2\pi^2\kappa t}{L^2}\right]\right\} \tag{13.1}$$

を Python により数値計算する方法を考える．ここで，与えるべき物理パラメータは，

(1) 熱拡散率 κ

(2) 密度 ρ

(3) 定圧比熱 C_p

(4) マントルあるいはプレートの厚さ L

(5) プレートの年代 t

である．さらに，温度を計算する点の数 m_z も必要である．点の間隔 dz は，

$$dz = \frac{L}{m_z} \tag{13.2}$$

であり，i 番目の点の深さは

$$z = dz \times i \tag{13.3}$$

と計算できる．

　プログラムの中身はあとで説明するが，プログラムは以下のような処理の流れになっている．

(1) パラメータをファイルから読み込む

(2) k や dz を計算

(3) i-方向(深さ方向)への繰り返し

(4) フーリエ級数部分の計算：サブルーチン

(5) 温度の計算

(6) 深さと温度を出力する

13.2 Jupyter Notebook によるプログラムの実行

プログラムを Jupyter Notebook に入力して，シミュレーションを行う方法を説明する。

(1) Windows のファイルブラウザ(Explorer)からプログラム格納用フォルダの作成を作成する.

例えば，フォルダ書類フォルダなどの中に新しいフォルダ“Python”などを作る.

(2) 12 章同様(140 行目)に Jupyter Notebook を開く。

(3) Documents→Python をクリックして書類フォルダの中にある Python フォルダに入る。

(4) 右上の New→Text File を選ぶ

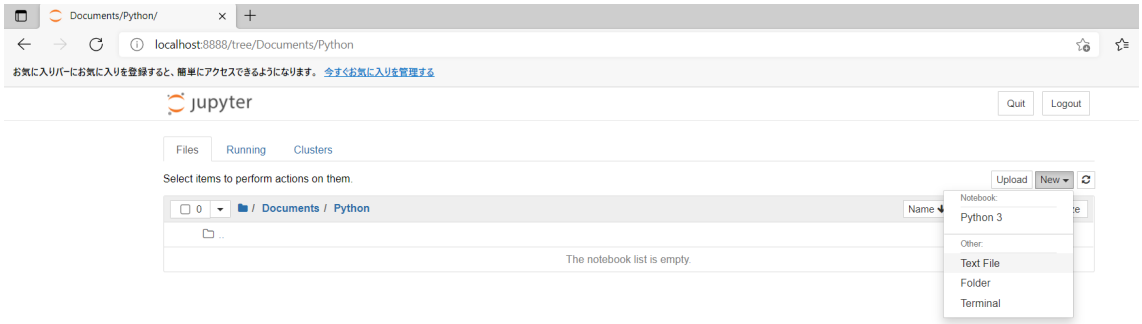

図 13.1 新規テキストファイルの作成

(5) 選ぶと，新しいタブが開いてプログラムやデータを入力するエディタ画面となる。ただし，テキストファイルを作成する作業は TeraPad やメモ帳など他のテキストエディタで行っても構わない。

(6) 81 ページ以降にあるプログラムリストを見てモジュール (プログラム)あるいはデータファイルを入力する。

(7) 上にある Untitled.txt をクリックする。

図 13.2 新規テキストファイルの入力画面

(8) Rename file というプロンプトがでるので，ファイルにはそれぞれ，次のような名前をつける。

plcm_main.py

(9) 拡張子が.py に変わると，Python のファイルと認識されるので，予約語などの色が変わる。

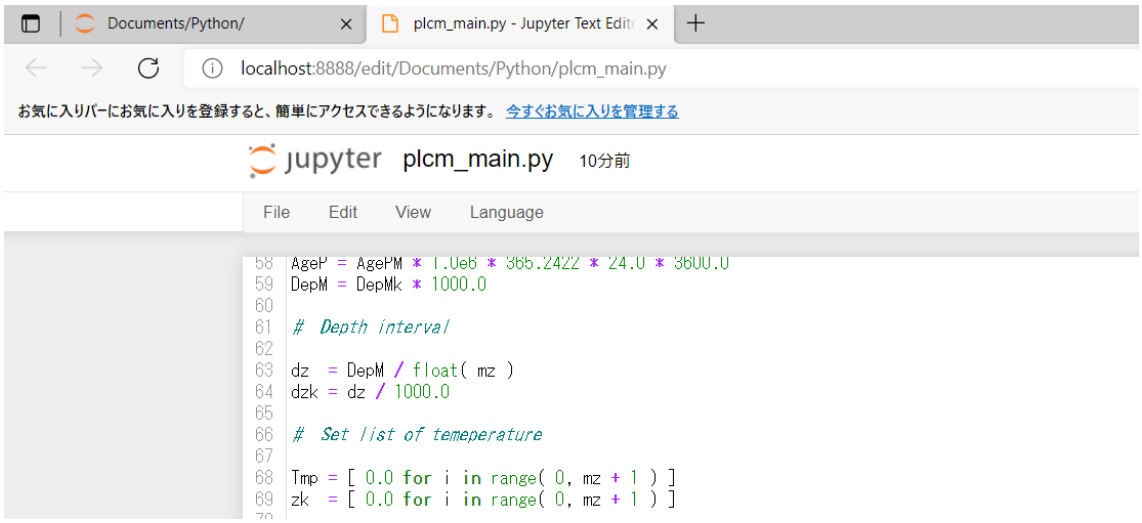

図 13.3 ファイル名が変更された後

(10)最初のタブに戻る

(11)右上の New→Text File を選び，残りの２つのファイルも入力する。

(12)上にある Untitled.txt をクリックする。

(13)Rename file というプロンプトがでるので，ファイルにはそれぞれ，次のような名前をつける。

`plcm_sineexp.py`

`plcm_para.data`

(14)右上の Logout をクリックしたあと，ブラウザを閉じて Jupyter Notebook を終了する。

(15)Jupyter Notebook を再度起動する。

(16)Documents→Python をクリックして書類フォルダの中にある Python フォルダに入る。

(17)右上の New→Python 3 を選ぶ

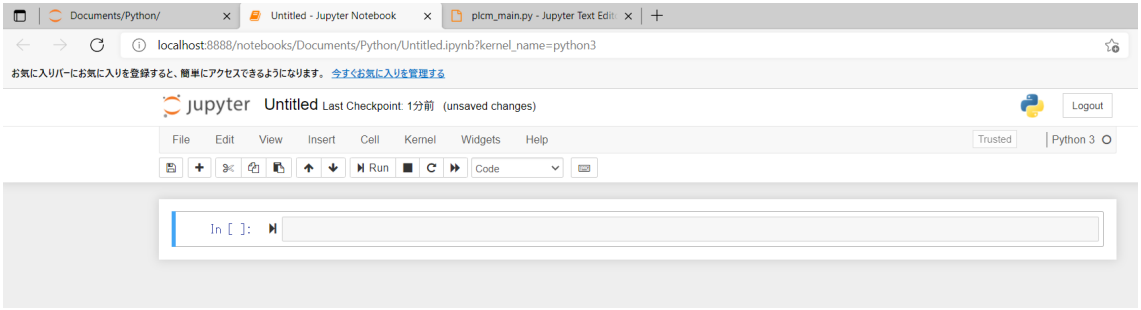

図 13.4 Jupyter Notebook の Python 開発画面

(18)メインモジュールを編集していたタブで，プログラムを全てコピーしてから，Untitled をクリックして適当な名前に変更する。

(19)Run ボタンを押す。

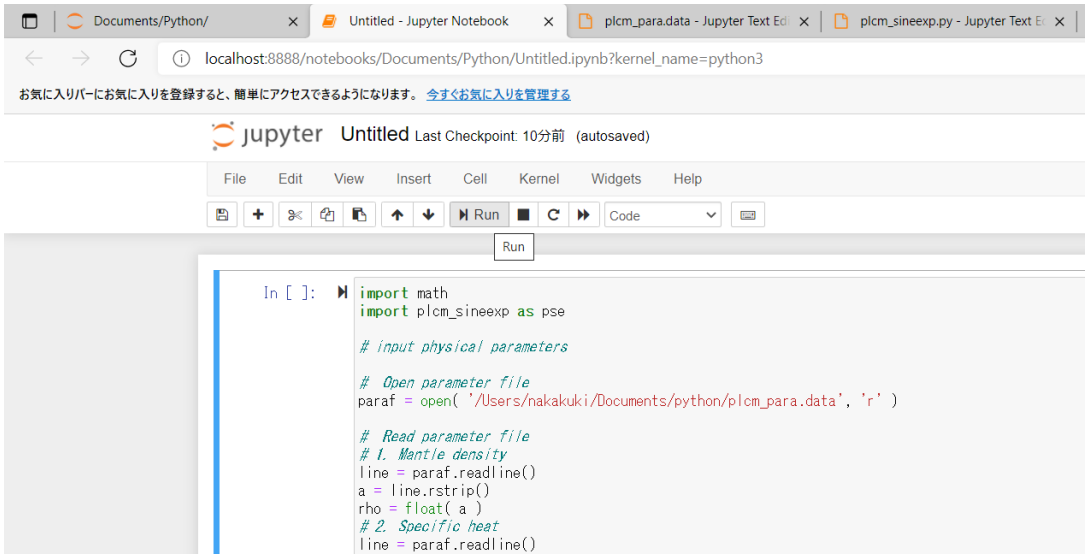

図 13.5 プログラムの実行

(20)実行されると次のようになる。計算結果は `plcm_T_080.csv` というファイルに出力される。

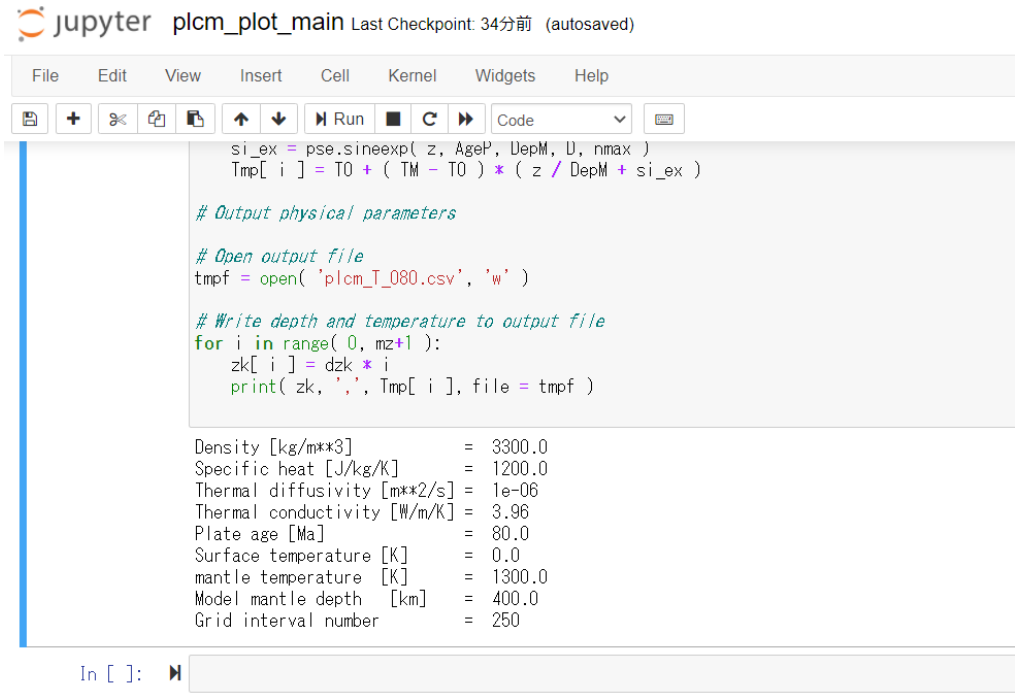

図 13.6 プログラムの実行後の表示．読み込んだ物理パラメータが表示される

(21)プレートの年代を変更するときには plcm_para.data のプレートの年代とプログラム地位にある出力ファイルの名前を変更してから実行する。

13.3 Jupyter Notebook によるグラフ表示

プログラムを Jupyter Notebook を用いてグラフを表示させる。

(1) プログラムの上部にグラフモジュールのインポートに関する次の行を加える。

```
import matplotlib.pyplot as plt
```

ここでは，matplotlib.pyplt 関数を plt という引用で呼び出すという意味である。

(2) プログラムにグラフをプロットする関数の呼び出しをする行を加える。

```
#  plot
plt.plot( zk, Tmp )
plt.grid( color = '0.8' )
plt.show
```

(3) グラフが表示される。

```
Density [kg/m**3]            =  3300.0
Specific heat [J/kg/K]       =  1200.0
Thermal diffusivity [m**2/s] =  1e-06
Thermal conductivity [W/m/K] =  3.96
Plate age [Ma]               =  80.0
Surface temperature [K]      =  0.0
mantle temperature  [K]      =  1300.0
Model mantle depth   [km]    =  400.0
Grid interval number         =  200
Out[3]: <function matplotlib.pyplot.show(*args, **kw)>
```

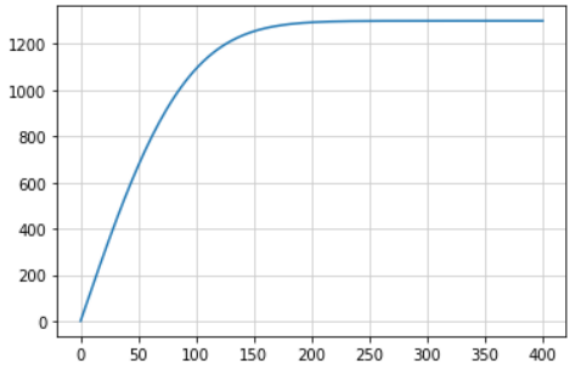

図 13.6 表示されたグラフ.

プログラムリスト

ここにあげるプログラムリストは python インタープリター上で動作するものである。JupyterNotebook でも動作させることができるが，その場合には，エディタの部分へコピーペーストする必要がある。さらに，JupyterNotebook のグラフィック機能を利用したい場合は，ここにあるリストの内容だけでなく，モジュールのインポートおよびグラフィック関数の呼び出しの部分を書き加える必要がある。プログラムの書かれたモジュールファイル２つのほか，パラメータを入力するためのデータファイルが必要である．それぞれのファイルを同じディレクトリに置いて，実行すること．

メインモジュール plcm_main.py

```
import math
import plcm_sineexp as pse

# input physical parameters

#  Open parameter file
paraf = open( 'plcm_para.data', 'r' )

#  Read parameter file
# 1. Mantle density
line = paraf.readline()
a = line.rstrip()
rho = float( a )
# 2. Specific heat
line = paraf.readline()
a = line.rstrip()
Cp = float( a )
# 3. Thermal diffusivity
line = paraf.readline()
a = line.rstrip()
D = float( a )
# 4. Plate age
```

```
346   line = paraf.readline()
347   a = line.rstrip()
348   AgePM = float( a )
349   # 5. Surface temperature
350   line = paraf.readline()
351   a = line.rstrip()
352   T0 = float( a )
353   # 6. Mantle temperature
354   line = paraf.readline()
355   a = line.rstrip()
356   TM= float( a )
357   # 7. Model mantle depth
358   line = paraf.readline()
359   a = line.rstrip()
360   DepMk = float( a )
361   # 8. Grid interval number
362   line = paraf.readline()
363   a = line.rstrip()
364   mz = int( a )
365
366   # Thermal conductivity
367   k   = rho * Cp * D
368
369   print( "Density [kg/m**3]            = ", rho )
370   print( "Specific heat [J/kg/K]       = ", Cp )
371   print( "Thermal diffusivity [m**2/s] = ", D )
372   print( "Thermal conductivity [W/m/K] = ", k )
373   print( "Plate age [Ma]               = ", AgePM )
374   print( "Surface temperature [K]      = ", T0 )
375   print( "mantle temperature  [K]      = ", TM )
376   print( "Model mantle depth   [km]    = ", DepMk )
377   print( "Grid interval number         = ", mz )
378
```

```
379  #  Convert unit km -> m, My -> Sec
380
381  AgeP = AgePM * 1.0e6 * 365.2422 * 24.0 * 3600.0
382  DepM = DepMk * 1000.0
383
384  #  Depth interval
385
386  dz  = DepM / float( mz )
387  dzk = dz / 1000.0
388
389  #  Set list of temeperature
390
391  Tmp = [ 0.0 for i in range( 0, mz + 1 ) ]
392  zk  = [ 0.0 for i in range( 0, mz + 1 ) ]
393
394  #    print( i, Tmp[ i ] )
395
396  #  maximum degreeof Fourier series
397
398  nmax = int( mz / 2 )
399
400  #  Calculate temperature
401
402  for i in range( 0, mz + 1 ):
403      z = dz * i
404      si_ex = pse.sineexp( z, AgeP, DepM, D, nmax )
405      Tmp[ i ] = T0 + ( TM - T0 ) * ( z / DepM + si_ex )
406
407  # Output physical parameters
408
409  # Open output file
410  tmpf = open( 'plcm_T_080.csv', 'w' )
411
```

```
412 # Write depth and temperature to output file
413 for i in range( 0, mz+1 ):
414     zk[ i ] = dzk * i
415     print( i, ',', zk[ i ], ',', Tmp[ i ], file = tmpf )
416
417
```

418 関数モジュール plcm_sineexp.py

```
419
420 def sineexp( z, AgeP, DepM, D, nmax ):
421     import math
422
423 #  pi
424     pi = math.pi
425 #  set list f
426     fe  = [ 0.0 for n in range( 0, nmax + 1 ) ]
427
428 #  n-degree term of Fourier series multiplied by dumping factor due to thermal
429 conduction
430     for n in range( 1, nmax + 1 ):
431         si = math.sin( n * pi * z / DepM)
432         en = math.exp( - ( n * pi / DepM )**2 * D * AgeP )
433         fe[ n ] = 2.0 / ( n * pi ) * si * en
434
435 #  make series by summation from n = 1 to nmax
436     sumf = 0.0
437     for n in range( 1, nmax + 1 ):
438         sumf = sumf + fe[ n ]
439
440     siex = sumf
441
442     return siex
443
444
```

パラメータデータファイル plcm_para.data

```
3300.0
1200.0
1.0e-6
80.0
0.0
1300.0
400.0
200
```

プログラム解説

１次元熱伝導方程式の数値解を求めるプログラムである．入力されたインターバルで深さを変えながら温度を計算する．プログラムは２つのモジュールに分けられており，フーリエ級数を含む級数を計算する部分をサブモジュール，その他をメインモジュールにしている．それぞれを１つのファイルになっている．

メインモジュール

`import math`： 数学関数のモジュールをインポート．

`import plcm_sineexp as pse`：自分で定義した級数を計算する関数モジュール `plcm_sineexp` を `pse` という名前でインポートする．モジュールの定義が含まれるファイルの名前は `plcm_sineexp.py` としなければならない．

`paraf = open( '/Users/nakakuki…`：`open` 関数．`plcm_para.data` を `paraf` というオブジェクト(ファイルオブジェクト)に関連づける．`'r'`はファイルを読み込むことを意味する。

`line = paraf.readline()`：ファイルオブジェクト `paraf` から１行読み込み変数オブジェクト `line` に代入．`.readline()`ようにオブジェクトには特有の機能があり，機能を呼び出す方法をメソッドという．なお，Python ではデータは文字列データとして読み込まれる。データの部分だけでなく，改行コードも文字列として読んでしまう。

`a = line.rstrip()`：変数オブジェクト `line` に含まれる改行コードを削って文字だけにするメソッド。１行目だと，`3300.0\n` が `3300.0` になる。

`rho = float( a )`：文字変数 `a` の内容を浮動小数に変換して． `rho` に代入する．`float()`のような関数は型変換と呼ばれる。

print("Density…：出力関数で．ここでは文字列”Density …=”と rho の値が画面に出力される．

AgeP = AgePM *…：代入文．他の言語と同様，右辺の演算結果の値を左辺の変数に代入する．Python では左辺の変数の型は自動的に決まり，型宣言はない．

read (10,*) rho：入力文．ファイルの２行目の数値を倍精度実数変数 rho に読み込む．

write (6,*) apara,' = ',rho：出力文．apara の値と“=”と rho の値を装置番号 6 番の装置(標準出力装置：画面)に 1 行に出力する．

Tmp = [0.0 for i in range(0, mz + 1)]：リストオブジェクトを初期化するための代入文．右辺の[ ]で Tmp がリストオブジェクトであることが自動的に決定される．例えば，Tmp = [ ] と書くと空のリストが定義される．Python ではデータ列や行列をリストオブジェクトという．Fortran の配列変数に相当する機能である．i=0 から mz になるまで(mz+1 までではないことに注意！あと，i の値は絶対値としては意味を持っていない，つまりリストの最初にある値が必ず i=0 での値なってしまう)，0.0 を代入している．Python では変数オブジェクトは自動的に型が決まるので，初期化しておかないでリストオブジェクトを使用するとエラーになる．例えば，突然 Tmp[0] = 0.0 などと書くことはできない。

nmax = int(mz / 2)：代入文．関数 int()は整数型への変換の関数である．整数型変数どうしの四則演算は割り算を除いて結果が整数型となるが，割り算は自動的に実数型になってしまう．nmax を整数型としたいので，関数 int()を用いている．Python では Fortran のような変数の型宣言はなく，自動的に決定される．

for i in range(0, mz + 1):：繰り返し処理．i の値が 0 から mz までインデントしてある部分の処理を繰り返し行う．mz+1 までではないので，注意．

Tmp[i] = T0 + (TM - T0) * (z / DepM + si_ex)：代入文．リストオブジェクトを呼び出すときの添字はカギ括弧[ ]に入れて表す．温度 Tmp を計算している

tmpf = open('/Users/nakakuki…：open 関数．plcm_T_080.csv を tmpf というオブジェクト(ファイルオブジェクト)に関連づける．'w'はファイルに書き込む動作をすることを意味する。

print(i, ',', zk[i], ',', Tmp[i], file = tmpf)：出力関数．i，zk と Tmp[i]の値を file=で指定しているファイルオブジェクトに出力する．出力は 1 行ずつ行われ，行の後ろには改行コードが挿入される．','があるので，２つの値の間にはコンマが挿入される．マントルあるいはプレート底の点の温度を代入している．

関数モジュール

def sineexp(z, AgeP, DepM, D, nmax): ： def 文．関数モジュールを定義することを表す．次の行からはインデントによって関数モジュールの単位を表す．( )の中は引数が入

る．引数の型は呼び出し側で定義したものと同じになるが，それぞれのモジュール単位でローカルに定義される．呼び出される関数内でこれらの変数の値を変更することはできない． これは Fortran とは異なる点である．

`pi = math.pi`：代入文．右辺は数学関数 pi を呼び出している．

`for n in range( 1, nmax + 1 ):`：繰り返し処理．`i=1` から `nmax` までインデント内の処理を繰り返す．級数を計算するための繰り返しである．

`return siex`：return 文．siex の値を関数の値として呼び出し側に返す．

パラメータファイル

全ての行がデータとなっている．

上から，マントルの密度(kg m^{-3})，定圧比熱(J kg^{-1})，熱拡散率(m^2 s^{-1})，プレートの年代(Ma)，地表温度(°C)，マントルの温度(°C)，計算する最大の深さ(km)，計算点の間隔数(個)，出力ファイル名，である．