数値計算の基礎と地球惑星科学への応用

1

- 3 II. プレートの冷却の数値シミュレーション・その2
- 4 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
- 5 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.

6

7 10. Python を利用するための準備

- 8 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
- 9 Anaconda を導入する。

10

11 **10.1** Python

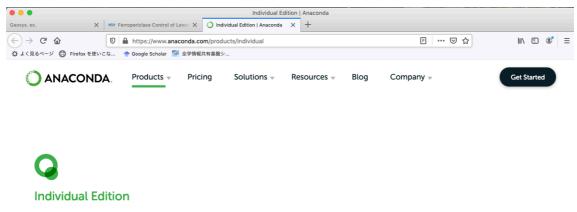
- 12 Python は高級プログラミング言語の1つで、文法が簡単なためにプログラミングの学
- 13 習に向いた言語である。人工知能の研究に利用されており、人工知能に関する多くの
- 14 ライブラリが開発されている。このためプログラミング入門用言語として人気が高
- 15 い。Python は次のような特徴を持つ。
- 16 (1) 高級プログラミング言語
- 17 (2) 開発者はグイド・ヴァンロッサム
- 18 (3) 文法を極力単純化してコードの可読性が高く書きやすい
- 19 (4) 標準ライブラリやサードパーティ製のライブラリ、関数など、さまざまな領域に特
- 20 化した豊富で大規模なツール群
- 21 (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
- 22 (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
- 23 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
- 24 (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
- 25 トウェア
- 26 (8) 現在, Python 2.7 と 3.7 の 2 つの系統

27

28 **10.2** Anaconda

- 29 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
- 30 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
- 31 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
- 32 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。

- 34 10.3 Anaconda のインストール
- 35 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
- 36 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
- 37 語のアカウントの場合は、英語のアカウントを用意する。
- 38 (1) 次の URL にアクセスする。
- 39 https://www.anaconda.com/products/individual



Your data science toolkit

41

42 **図 10.1** Anaconda 公式ダウンロードサイト

43

44 (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。

45

	Anaconda Installer	5
Windows #	MacOS É	Linux 🗴
Python 3.7	Python 3.7	Python 3.7
64-Bit Graphical Installer (466 MB)	64-Bit Graphical Installer (442)	64-Bit (x86) Installer (522 MB)
32-Bit Graphical Installer (423 MB)	64-Bit Command Line Installer (430 MB)	64-Bit (Power8 and Power9) Installer (276 MB)
Python 2.7	Python 2.7	
64-Bit Graphical Installer (413 MB)	64-Bit Graphical Installer (637 MB)	Python 2.7
32-Bit Graphical Installer (356 MB)	64-Bit Command Line Installer (409 MB)	64-Bit (x86) Installer (477 MB)

46

47 図 **10.2** 様々な OS に対応したインストーラ

48 (3) ダウンロードしたインストーラを起動する

49

.

Anaconda3 2020.02 (64-bit) Setup

Welcome to Anaconda3 2020.02 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 2020.02 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

O ANACONDA

50

図 10.3 Anaconda インストーラが起動

52

51

53 (4) デフォルトインストールで良いので[Next]を押し続ける

54

Anaconda3 2020.02 (64-bit) Setup — X

Select Installation Type

Please select the type of installation you would like to perform for Anaconda3 2020.02 (64-bit).

Install for:

All Users (requires admin privileges)

Anaconda, Inc.

Anaconda, Inc.

Anaconda, Inc.

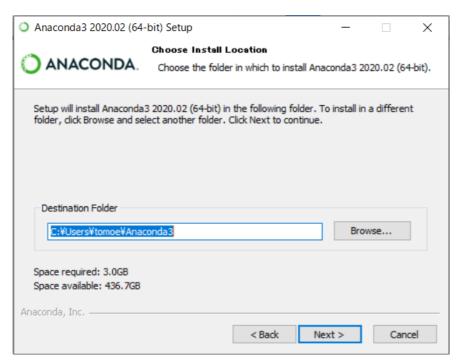
Anaconda, Inc.

Anaconda, Inc.

Cancel

55

図 10.4 Anaconda インストーラが起動



59 **図 10.5** Anaconda のインストール先選択・デフォルトで良い

61 (4) インストーラを終了する。

58

60

63

64

66

6970

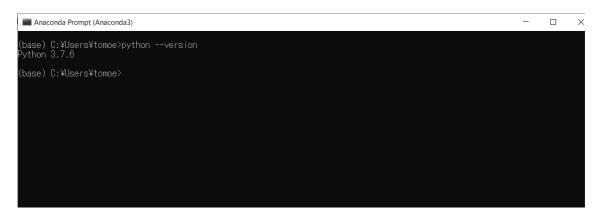
62 (5) ドキュメントフォルダの中の"Python Scrips"を確認。

□ | ▼ | ドキュメント ファイル ホーム 共有 表示 ← → ∨ ↑
□ > PC > ドキュメント ★ ^ □ 名前 更新日時 ➡ ダウンロード 状態 種類 サイズ □ ドキュメント fort \odot 2020/04/30 16:40 ファイル フォルダー Office のカスタム テンプレート 0 2020/04/15 14:12 ファイル フォルダー Python Scripts 2020/05/06 18:33 ファイル フォルダー ▮ 情報活用演習20% Screen_shots 0 2020/05/06 19:16 ファイル フォルダー ▶ 地球惑星内部物³ \odot 2020/05/04 11:12 ファイル フォルダー

65 **図 10.6** Anaconda のインストール先選択

67 (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 68 ることを確認。

- 71 10.3 インストール後のテスト
- 72 (1) [Anaconda3 (64bit)]→[Anaconda Powershell Prompt (Anaconda3)]をクリック
- 73 Powershell Prompt では Unix コマンドが使える
- 74 (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
- 75 (3) "python -version" と入力して python のバージョンを確認する



7778

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

7980

11. Python インタプリタ

8182

- 83 11.1 Python インタプリタの起動とプログラムの実行
- 84 (1) "python"と入力すると python インタープリタが起動する
- 85 (2) プロンプトが">>>"に変わる
- 86 (3) 次のように入力する。1行毎にエンターキーを押す
- $87 \implies a = 3$
- $88 \implies b = 1$
- 89 >>> c = a + b
- 90 >>> print (c)

```
Anaconda Powershell Prompt (Anaconda3)

(base) PS C:\text{Users\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{YUsers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yusers\text{Yuser\text{Yusers\text{Yusers\text{Yusers\text{Yusers\tex
```

93 図 11.1 Python インタプリタとプログラムの逐次入力・実行

94

92

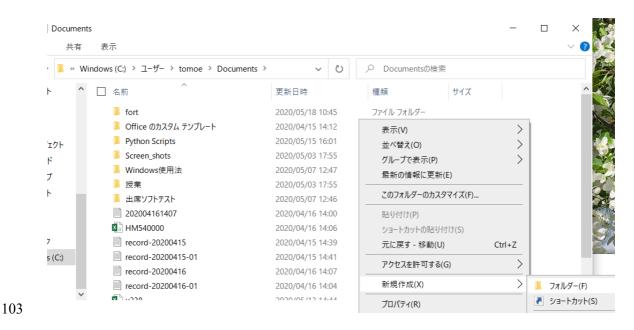
- 95 (4) 上のように計算結果が表示される
- 96 (5) [Ctrl]+[D]を押すと python インタプリタが終了する

97

98 11.1 プログラムの保存と保存したプログラムの実行

- 99 (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
- 100 [右クリック]→[新規作成]→[フォルダー]
- 101 例えば、C:Users¥*yourID*¥Documents¥python_prog など

102

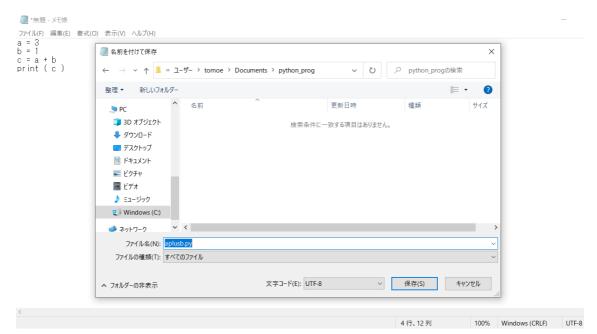


104 **図 11.2** Python プログラム保存先の作成

105106

(1) 「メモ帳」や「TerraPad」などのテキストエディタを起動する

- 107 「メモ帳」は[スタートメニュー]→[Windows アクセサリ]の中にある
- 108 (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。
- 109 (3) [名前をつけて保存]を選ぶ
- 110 (4) プログラム保存フォルダ C:Users¥yourID\DeltaDocuments\python_prog に移動する
- 111 (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
- 112 (5) ファイルの名前を入力, [保存]ボタンを押す



115 **図 11.3** Python プログラム保存先の作成

116

- 117 (6) 先ほど python インタプリタを動かした Anaconda PowerShell Prompt を前面に
- 118 持ってくる
- 119 (7) プロンプトに現在のフォルダがでているので、
- 120 ¥Users¥*yourID*
- 121 になっているか確認する
- 122 (8) 次のコマンドでフォルダを移動
- 123 > cd Documents\(\text{python_prog}\)
- 124 (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
- 125 確認
- 126 (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
- 128 (11) 結果が表示されているか確認する

130 図 11.4 プログラム確認と実行

131132

133

129

12. Jupyter Notebook 使用法

134

135 12.1 Jupyter Notebook の基本

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

137

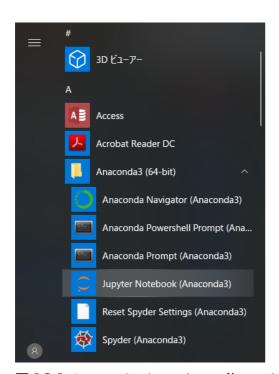
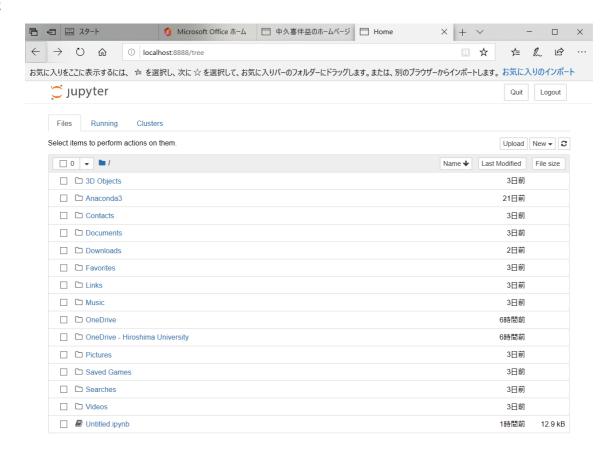


図 12.1 Anaconda インストール後のスタートメニュー

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

142



143

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

145

146 (3) 右の[New]から[Python 3]を選ぶ

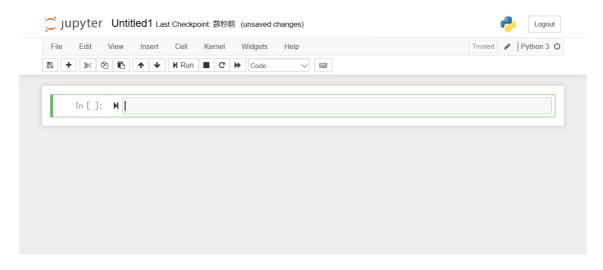
147



148

149 図 12.3 新しいプログラムの選択

150 (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。



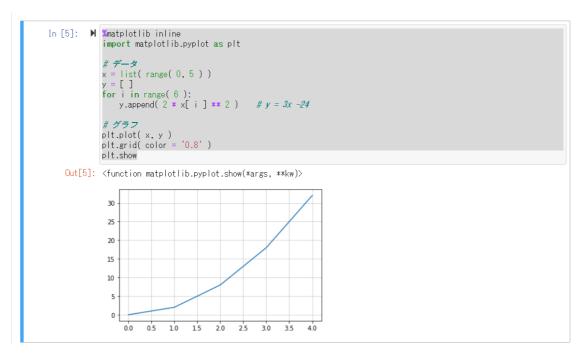
153 図 12.4 Jupyter Notebook が起動したブラウザ (Microsoft Edge)

155 (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。

158 図 12.5 プログラム入力中の表示

161 (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 162 が実行される。このプログラムではグラフが表示される。

163



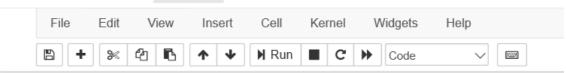
164

165 図 12.6 実行後の表示

166

167 (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168 [Rename]を選ぶ (名前をつけて保存ではない)。

Checkpoint: 5分前 (unsaved changes)



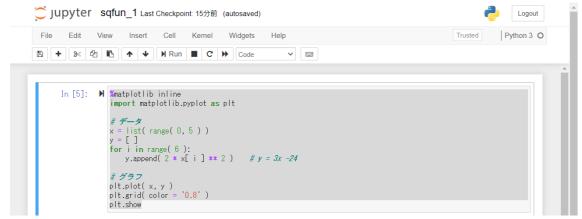
169

170 図 12.6 ファイルの保存

171

173 (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 174 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 175 ル名に python 拡張子.py つけてはいけない。自動的に.ipynb になる)。

176



178 図 12.7 ファイル保存後の表示

179

177

180 (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 181 ぶ (クリックする)。

182



図 12.8 Jupyter Notebook ファイルブラウザの表示

184185

186 **13. Python によるプレートの冷却の数値シミュレーション**

187 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.

188

189 **13.1 Python** プログラムによる数値計算

190 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

$$T_{i}(t) = T_{0} + (T_{M} - T_{0}) \left\{ \frac{z_{i}}{L} + \sum_{n=1}^{M} \frac{2}{n\pi} \sin\left[\frac{n\pi z_{i}}{L}\right] \exp\left[\frac{n^{2}\pi^{2}\kappa t}{L^{2}}\right] \right\}$$
(13.1)

- 192 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
- 193 は,

200

- 194 (1) 熱拡散率 κ
- 195 (2) 密度 ρ
- 196 (3) 定圧比熱 C_p
- 197 (4) マントルあるいはプレートの厚さ *L*
- 198 (5) プレートの年代 t
- 199 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、

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

201 であり, *i* 番目の点の深さは

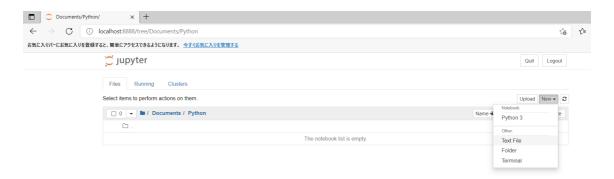
$$z = dz \times i \tag{13.3}$$

- 203 と計算できる.
- 204 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
- 205 っている.
- 206 (1) パラメータをファイルから読み込む
- 207 (2) k や dz を計算
- 208 (3) i-方向(深さ方向)への繰り返し
- 209 (4) フーリエ級数部分の計算:サブルーチン
- 210 (5) 温度の計算
- 211 (6) 深さと温度を出力する

- 213 13.2 Jupyter Notebook によるプログラムの実行
- 214 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
- 215 る。

- 217 (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
- 218 作成する.
- 219 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.
- 220 (2) 12 章同様(140 行目)に Jupyter Notebook を開く。
- 221 (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
- 222 入る。
- 223 (4) 右上の New→Text File を選ぶ

224



225

226 図 13.1 新規テキストファイルの作成

227

- 228 (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
- 229 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
- 230 ストエディタで行っても構わない。
- 231 (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
- 232 ータファイルを入力する。

234 (7) 上にある Untitled.txt をクリックする。

235

```
× *untitled.txt - Jupyter Text Editor × +
             C | localhost:8888/edit/Documents/Python/untitled.tx
                                                                                                                                                                                                              ô
お気に入りバーにお気に入りを登録すると、簡単にアクセスできるようになります。 <u>今すぐお気に入りを管理する</u>
                             🂢 Jupyter untitled.txt ᠀分前
                              File Edit View Language
                                                                                                                                                                                                      Plain Text
                                 | AgeP = AgePM * 1.0e6 * 365.2422 * 24.0 * 3600.0
| DepM = DepMk * 1000.0
                                 # Depth interval
                                 dz = DepM / float( mz )
dzk = dz / 1000.0
                                # Set list of temeperature
                                 Tmp = [ 0.0 for i in range( 0, mz + 1 ) ]
zk = [ 0.0 for i in range( 0, mz + 1 ) ]
                                  # print(i, Tmp[i])
                                 # maximum degreeof Fourier series
                                 nmax = int( mz / 2 )
                                  # Calculate temperature
                                 for i in range( 0, mz + 1 ):
z = dz * i
                                      si_ex = pse.sineexp( z, AgeP, DepM, D, nmax )
Tmp[ i ] = T0 + ( TM - T0 ) * ( z / DepM + si_ex )
```

237 図 13.2 新規テキストファイルの入力画面

238

236

- 239 (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
- 240 前をつける。
- 241 plcm_main.py
- 242 (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243 が変わる。

244

245

246 図 13.3 ファイル名が変更された後

247

248 (10)最初のタブに戻る

- 249 (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。
- 250 (12)上にある Untitled.txt をクリックする。
- 251 (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
- 252 名前をつける。
- 253 plcm_sineexp.py
- 254 plcm_para.data
- 255 (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
- 256 する。
- 257 (15)Jupyter Notebook を再度起動する。
- 258 (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
- 259 入る。

262

264

267

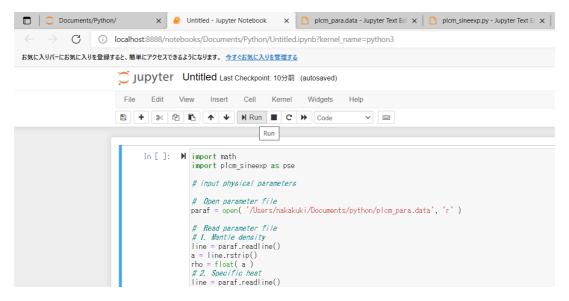
260 (17)右上の New→Python 3 を選ぶ

263 図 13.4 Jupyter Notebook の Python 開発画面

265 (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、
 266 Untitled をクリックして適当な名前に変更する。

268 (19)Run ボタンを押す。

269



271 図 13.5 プログラムの実行

272

270

273 (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 274 される。

275

```
ご Jupyter plcm_plot_main Last Checkpoint: 34分前 (autosaved)
 File Edit View
                        Insert Cell
                                         Kernel
                                                 Widgets
~
                       si_ex = pse.sineexp( z, AgeP, UepM, U, nmax )
Tmp[ i ] = TO + ( TM - TO ) * ( z / DepM + si_ex )
                   # Output physical parameters
                   # Open output file
                   tmpf = open( 'plcm_T_080.csv', 'w' )
                    # Write depth and temperature to output file
                   for i in range( 0, mz+1 ):
                       zk[ i ] = dzk * i
print( zk, ',', Tmp[ i ], file = tmpf )
                   Density [kg/m**3]
                                                 = 3300.0
                   Specific heat [J/kg/K] =
Thermal diffusivity [m**2/s] =
Thermal conductivity [W/m/K] =
                                                 = 1200.0
                                                     1e-06
                   Plate age [Ma]
                                                    80.0
                   Surface temperature [K]
                                                    0.0
                   mantle temperature [K]
                                                 = 1300.0
                   Model mantle depth [km]
                                                 = 400.0
                   Grid interval number
                                                  = 250
      In [ ]: ₩
                                                                                                        289
```

290 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

291 (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 292 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。

293

294

13.3 Jupyter Notebook によるグラフ表示

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

296

- 297 (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。
- 298 import matplotlib.pyplot as plt
- 299 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
- 300 る。
- 301 (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。
- 302 # plot
- 303 plt.plot(zk, Tmp)
- 304 plt.grid(color = '0.8')
- 305 plt.show

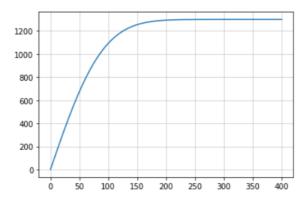
306

307 (3) グラフが表示される。

308

```
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
                               1300.0
mantle temperature [K]
                                400.0
Model mantle depth [km]
Grid interval number
                                200
```

Out[3]: <function matplotlib.pyplot.show(*args, **kw)>



309310

図 13.6 表示されたグラフ.

311

313 プログラムリスト

- 314 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。
- 315 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
- 316 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、
- 317 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
- 318 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、
- 319 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
- 320 クトリに置いて、実行すること.

321

322 メインモジュール plcm_main.py

323

- 324 import math
- 325 import plcm_sineexp as pse

326

327 # input physical parameters

328

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

- 332 # Read parameter file
- 333 # 1. Mantle density
- 334 line = paraf.readline()
- 335 a = line.rstrip()
- 336 rho = float(a)
- 337 # 2. Specific heat
- 338 line = paraf.readline()
- 339 a = line.rstrip()
- 340 Cp = float(a)
- 341 # 3. Thermal diffusivity
- 342 line = paraf.readline()
- 343 a = line.rstrip()
- 344 D = float(a)
- 345 # 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()
- $352 \quad T0 = float(a)$
- 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()
- $360 \quad \text{DepMk} = \text{float(} a \text{)}$
- 361 # 8. Grid interval number
- 362 line = paraf.readline()
- $364 \quad mz = int(a)$
- 365
- 366 # Thermal conductivity
- $367 \quad k = \text{rho} * \text{Cp} * \text{D}$
- 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)

```
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 \text{ for i in range}(0, mz + 1)]
392
      zk = [0.0 \text{ 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
      tmpf = open( 'plcm_T_080.csv', 'w' )
410
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 \text{ for n in range}(0, \text{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
          for n in range( 1, nmax + 1 ):
437
438
              sumf = sumf + fe[ n ]
439
440
          siex = sumf
441
442
           return siex
443
444
```

445	パラメータデータファイル plcm_para.data
446	
447	3300.0
448	1200.0
449	1.0e-6
450	80.0
451	0.0
452	1300.0
453	400.0
454	200
455	
456	
457	プログラム解説
458	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
459	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
460	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
461	を1つのファイルになっている.
462	
463	メインモジュール
464	import math: 数学関数のモジュールをインポート.
465	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
466	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
467	plcm_sineexp.py としなければならない.
468	paraf = open('/Users/nakakuki…:open 関数. plcm_para.dataを paraf というオブジェク
469	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。
470	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
471	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
472	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
473	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。
474	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
475	ソッド。1行目だと,3300.0\n が3300.0 になる。
476	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

477 うな関数は型変換と呼ばれる。

- 478 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
- 479 る.
- 480 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.
- 481 Python では左辺の変数の型は自動的に決まり、型宣言はない.
- 482 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.
- 483 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
- 484 準出力装置:画面)に1行に出力する.
- 485 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
- 486 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,
- 487 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
- 488 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
- 489 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
- 490 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
- 491 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
- 492 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。
- 493 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
- 494 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
- 495 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
- 496 ような変数の型宣言はなく、自動的に決定される。
- 497 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
- 498 分の処理を繰り返し行う. mz+1 までではないので,注意.
- 499 Tmp[i] = T0 + (TM T0) * (z / DepM + si_ex):代入文. リストオブジェクトを呼び
- 500 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
- 501 tmpf = open('/Users/nakakuki…: open 関数. plcm_T_080.csv を tmpf というオブジェクト(フ
- 502 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。
- 503 print(i,',', zk[i], ',', Tmp[i], file = tmpf): 出力関数. i, zkと Tmp[i]の値
- 504 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
- 505 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.
- 506 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.
- 508 関数モジュール

- 509 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
- 510 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

511 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 512 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 513 い. これは Fortran とは異なる点である. 514 pi = math.pi:代入文. 右辺は数学関数 pi を呼び出している. 515 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. 516 return siex: return 文. siexの値を関数の値として呼び出し側に返す. 517 518 519 パラメータファイル 520 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 m³), 定圧比熱(J kg¹), 熱拡散率(m² s¹),プレートの年代(Ma), 521 522 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 523 イル名,である. 524

525

526