Pythonで天文計算

天文計算

太陽は日々、規則正しく動きながらも、日の出・日の入りの時刻は季節とともに変化していきます。夜空に見える星たちも、日々、少しずつ移ろいでいきます。明るい惑星が見えることもありますし、時には国際宇宙ステーションが上空を通過する様子を見ることができたりします。日食や月食が起こる日はあらかじめ分かっており、その日が来るのを心待ちにしている方も多いでしょう。

このような種々の天文現象がいつ、どのように起こるのか、この冊子の読者の中には、天文暦を見たり、国立天文台などのホームページを検索して調べたりされる方も多いと思います。あるいは、科学館で発行している「こよみハンドブック」を利用されている方もおられるかもしれません。

図1　こよみハンドブック

そんな時、これらの現象を、自分で計算して予想してみたいと思ったことはないでしょうか。でも、天文現象の計算は難しい数学が必要で、とても無理だと諦めているかもしれません。

しかし近年は、いろいろなプログラミング言語が手軽に使えるようになっています。上記「こよみハンドブック」では、毎日の日の出・日の入り時刻の計算に、Excelに搭載されているVBA（Visual Basic for Applications）という言語を用いて計算しています。他にもいろいろなプログラミング言語がありますが、特に最近人気があるのがPythonという言語です。Pythonはオープンソースと呼ばれる言語で、誰でも無料で使うことができます。科学技術計算が得意なため、情報系のエンジニアの方々に広く使われています。近ごろ話題の機械学習のプログラムを書く際の言語として利用されており、ディープラーニングによる人工知能の技術開発の際にも使用されています。

またPythonでは、世界中の人たちによって、ライブラリと呼ばれる様々な用途のプログラムが作られているため、これらを利用することができます。天文計算専用の便利なライブラリもあるので、自分で計算式を作らなくても、日の出・日の入り等の暦の計算や、恒星・惑星の位置などを簡単に求めることができます。さらに、図形を描画するライブラリもあるので、計算結果を自動的に星図にプロットすることも容易に行うことができます。

そこで今回は、Pythonを使って天文計算をする方法をご紹介したいと思います。

準備

Pythonは、Windows、Mac、Linuxいずれのパソコンでも利用可能です。自分のパソコンで使う際には、Pythonの公式サイトからプログラムをダウンロードしてインストールすることになりますが、ここではインストール方法の詳細は省略します。

なお、実際にPythonを使う際には、統合開発環境（IDE）というものがあると便利です。統合開発環境とは、プログラムのソースコードを書く部分、コンパイラと呼ばれるソースコードを変換する部分、実行結果を表示する部分などをひとまとめにしたソフトです。Pythonをプログラミングするための統合開発環境としては、Anacondaというソフトがよく使用されます。Anacondaのダウンロードサイトからプログラムをダウンロードしてインストールするだけで、簡単に統合開発環境が整います。

（Anacondaダウンロードサイト：https//www. anaconda. com/）

またAnacondaを使えば、追加パッケージのインストールやバージョンの管理も簡単に行うことができます。Anacondaをインストールした時点で、主要なライブラリも同時にインストールされますが、次節で述べる天文計算ライブラリのPyEphemはインストールされていないので、パッケージの管理画面から追加でインストールを行います。

実際にプログラムを入力したり、実行したりする際は、Anacondaをインストールすると同時にインストールされるSpyderというソフトを使います。次節のプログラムは、Spyderを使ってプログラムの入力、実行、結果の確認を行うことができます。

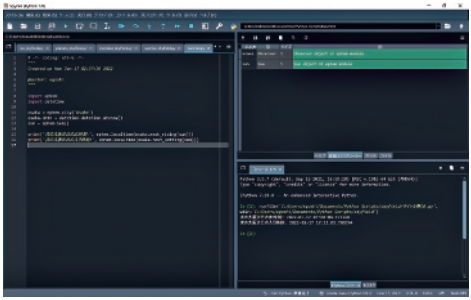

図2　Spyderの画面

日の出・日の入りの計算

ではインストールができたところで、早速、日の出・日の入りの時刻を計算してみることにします。

Pythonの天文計算ライブラリの1つとして、PyEphemというものがあります。このライブラリを使用することで、簡単に日の出・日の入り時刻の計算ができます。ちなみに英語でephemerisは天体暦を意味します。

実際のプログラムは次のようになります。

```
import   ephem
import   datetime
```

```
osaka = ephem.city('Osaka')
osaka.date = datetime.datetime.utcnow()
sun = ephem.Sun()

print('次の大阪の日の出時刻:',
      ephem.localtime(osaka.next_rising(sun)))
print('次の大阪の日の入り時刻:',
      ephem.localtime(osaka.next_setting(sun)))
```

わずか7行のプログラムですが、これだけで日の出・日の入り時刻を計算することができます。

このプログラムを実行してみると、次のように表示されます。

```
次の大阪の日の出時刻：2022-01-17 07:04:04.173568
次の大阪の日の入り時刻：2022-01-17 17:12:03.780374
```

ちゃんと日の出・日の入りの時刻を計算することができました。

では、プログラムの内容を見ていきましょう。最初の2行

```
import ephem
import datetime
```

は、必要なライブラリをインポートする部分です。1行目でPyEphemライブラリをインポートしています。2行目のdatetimeは、日付や時刻に対してさまざまな操作をすることができるライブラリです。

次に、観測者となる変数（観測者クラスのインスタンス：ここではosakaという名前にします）を作成します。大阪の場合は簡単で、あらかじめ都市名が登録されているため、次のようにcity('Osaka')として代入すると、大阪の緯度・経度が設定された観測者が作成されます。

```
osaka = ephem.city('Osaka')
```

通常は観測者の緯度と経度を直接記載して設定します。この場合は次のように、1行目でまず観測者（osaka）を作成し、2行目、3行目で作成した観測者に緯度と経度を設定します。PyEphemでは、緯度、経度は文字列として設定します。

```
osaka = ephem.Observer()
osaka.lat = '34.6914'
osaka.lon = '135.4917'
```

次に作成した観測者osakaに、現在の日時を設定します。日時の設定はdatetimeモジュールを使います。Pyephemで使用する時刻は、世界時で設定する必要があります。datetimeモジュールはutcnow()メソッドを使うと、現在の世界時での日付・時刻を返してくれます。これを先ほど作成した観測者osakaに代入します。

```
osaka.date = datetime.datetime.utcnow()
```

次に、太陽のインスタンス(ここではsunという名前にします)を作成します。これも、あらかじめ太陽を含め多くの天体について基本情報が登録されたクラスが準備されているため、次のようにするだけで設定できます。

```
sun = ephem.Sun()
```

これで準備は完了です。osakaという変数(クラス)には、緯度・経度・時間が設定され、sunという変数(クラス)が準備されました。

ここで、osakaに対してnext_rising(sun)というメソッドを実行すると、現在時刻から見て、一番早い次の日の出の時刻を計算してくれます。その結果は組み込み関数のprint()を使うと表示されます。

```
print(osaka.next_rising(sun))
```

この結果、次のように日の出時刻が表示されます。

```
2022-01-16 22:04:04
```

夜の10時に日の出？と思うかもしれませんが、この時刻は世界時です。PyEphemにはちゃんと、日本標準時に変換するメソッドもあり、localtime()メソッドを使用することで、日本標準時が得られます。

```
print(ephem.localtime(osaka.next_rising(sun)))
```

この結果、めでたく

```
2022－01－17　07:04:04
```

と、大阪の日の出時刻が得られました。

同様にnext_setting(sun)メソッドを用いれば、日の入り時刻を得ることができます。

クラスやメソッドの使い方が慣れないと難しく感じると思いますが、この形式は他の多くのプログラミング言語でも使われているものです。作成したクラスのインスタンスに緯度や経度など必要な値を設定し、目的のメソッドを用いて計算結果を得る、という流れになっています。

このように、天文計算の知識がなくても専用のライブラリを用いることで、世界各地、一年中どの日でも、日の出・日の入りの時刻を知ることができるのです。

日の出同時線

前節の方法を用いれば、特定の地点に関して、日の出・日の入り時刻を計算することができます。

一方、日本列島全体などの広い範囲について、日の出時刻の全般的な様子を知りたいという場合もあります。中でも初日の出は特別で、日本列島の中でどこが一番早いか、各地の日の出時刻が何時になるかを知りたいという方が多くおられます。これを一目で分かるようにしたのが、図3のような日の出の同時線と呼ばれるものです。

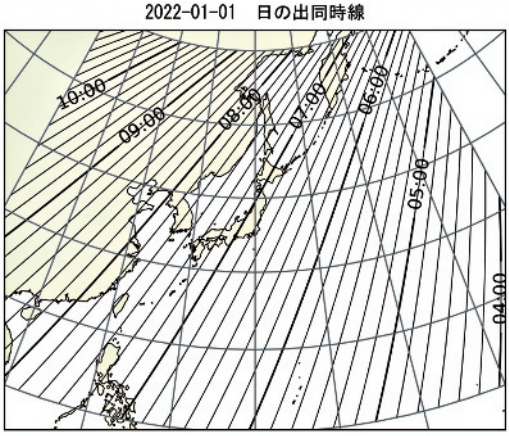

図3　初日の出同時線

元旦の頃は、北海道の根室よりも千葉の銚子の方が、わずかに日の出が早いことや、大阪の初日の出は7時5分ごろであることなどがすぐに分かります。

これもやはり、Pythonを用いて作成した図です。天文計算ライブラリであるPyEphemに加えて

・グラフ描画のためmatplotlibライブラリ
・地図描画のためcartopyライブラリ

を使用しています。

詳細な手法については長くなるので省略しますが、一言でいうと、前節の手法で格子状にいろいろな地点について日の出時刻を計算し、得られた結果をもとに、等高

線図を作成することで描画しています。

日の出とは？

通常、暦に書かれていたり、計算で得られる太陽の位置は、太陽の中心の位置です。一方、私たちが日の出と言えば、太陽の上端が地平線に見えた瞬間のことを指す場合が多いと考えられます。太陽の上端が地平線に現れた時、太陽の中心はまだ地平線の下にあります。太陽の上端が地平線に現れてから、太陽の中心が見えるまでには、およそ1分の時間がかかります。そのため日の出の時刻を計算する場合は、太陽の大きさを考慮する必要があります。

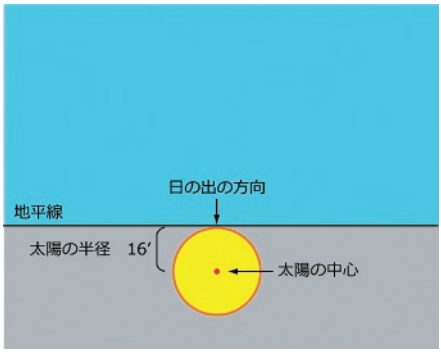

図4　日の出の瞬間

さらにややこしいことに、大気差というものがあります。これは、大気を通り過ぎる際に光が屈折するため、大気がない場合よりも、太陽が浮き上がって見えるという効果です。大気差によって、地平線付近では太陽はその直径分ほど浮き上がって見えます。そのため計算上の太陽は地平線よりも下にあるのに、実際には浮き上がりの効果で、日の出が見えるということになります。

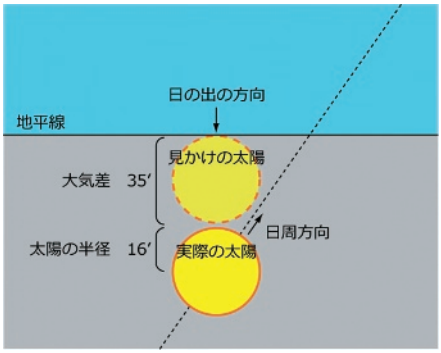

図5　大気差

実はPyEphemの計算で得られる日の出の時刻は、これらの効果をちゃんと考慮した値を計算してくれます。もし、さらにさかのぼって、太陽の位置から日の出の時刻を計算したいという場合は、これらの効果も考慮する必要があります。

Pythonのライブラリを使えば、工夫次第でほかにも任意の時刻の星図をプロットしたり、国際宇宙ステーションの通過予報、日食や月食の計算など、いろいろな計算を行うことができます。私のホームページでも、いくつか計算例を紹介しています。

（http://www.sci-museum.kita.osaka.jp/~egoshi/）

興味のある方は、ぜひいろいろな計算にチャレンジしてみてください。

江越　航（科学館学芸員）