2018년 6월 18일

제 1 절 Python

1.1 Python 소개

파이썬(Python)은 1991년 프로그래머인 귀도 반 로섬(Guido van Rossum)이 발표한 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑(dynamically typed) 대화형 언어이다. 파이썬이라는 이름은 귀도가 좋아하는 코미디 ⟨Monty Python's Flying Circus⟩에서 따온 것이다.

파이썬은 비영리의 파이썬 소프트웨어 재단이 관리하는 개방형, 공동체 기반 개발 모델을 가지고 있다. C언어로 구현된 C파이썬 구현이 사실상의 표준이다.

파이썬은 무료이며 누구나 다운받아 사용 가능 하다.

1.2 Python 설치

파이썬은 무료이며 누구나 다운받아 사용 가능 하다.

1.3 Python 시작하기

The Python programming language is a very popular, versatile, and (with its many extensions) very powerful tool. An interpreter that runs scripts in the programming language Python is freely distributed, and it may already be installed on your computer. There are many excellent free on-line tutorials on Python, including https://docs.python.org/ and Coursera classes beginning with https://www.coursera.org/learn/python. This class assumes that you have access to a tutorial or documentation, and will demonstrate how you can use the powers of Python to simulate things that happen in the natural world. The only way to learn a programming language is by doing something with it; just reading about it, it doesn't sink in in the same way. This class provides an opportunity for someone new to coding to get started, by providing detailed instructions for building a series of simple models applicable to the climate sciences. I've tried to write the instructions as if they were to go with some kind of kit, like to build a toy sailboat. The instructions are meant to make it as clear and easy as possible to succeed. By doing so, you will learn a lot about climate and Python both!

Using Python for numerical computation requires extensions which you will probably have to install. First is a numerical module called numpy (pronounced num-pie), and second, you'll also need a plotting module called matplotlib. On my Macintosh, I succeeded in using a package called anaconda (https://www.continuum.io/downloads), by installing their minimum package they call miniconda, and using that to install numpy and matplotlib. At the beginnings of your python scripts you will need to start these up using lines

Code 1.1

```python

import numpy
import matplotlib
```

where you should replace modelname.py with whatever the name of your script file is.

1.4 Editing Files

To create and edit Python scripts or Fortran source files, you will need to find or install an editor which can write plain text files. Word processors such as Word or TextEdit may be able to save clean text files, if you specifically look for that. A way to check is to type “more <filename>” in a terminal window, where you substitute the name of the file you want to inspect for <filename>, which will give you a screen-full of the file at a time. If it looks like plain letters of python text, you’re good. Examples of formats which won’t work include files with the suffix .doc, .docx, .rtf, and .pdf.

Alternatively, you may find some integrated programming environment for your computer that shows you the editable python source code in one part of a window, and output in another part.

1.5 Getting your code to work

Create your first script by copying pieces from some example script. When you first try to run it, it probably will have some syntax errors that will prevent the Python interpreter from working. The interpreter will tell you what line it decided it had to give up, which is usually the line where the error is, but not always. It can be a problem at an earlier line, like a mismatched parenthesis in a line above. Sometimes the only way to find where a problem line is, is to temporarily delete or comment out whole blocks of a code, until it starts running. Google is your friend here; you can copy the entire error message into the Google search box, add the word python or anything else relevant to what you’re trying to do, and probably you will find posts where the same question has been asked and answered. In particular, a web site called stackoverflow.com is a treasure chest of helpful information.

Once the code gets correct enough to run, it will probably give wrong answers that will require debugging. The simplest way to probe the numbers that the code is creating is to put in temporary print statements, printing out values of variables. Another fancier option is to use a debugger, which allows you to stop the code at some line number, step forward a bit at a time, and ask it interactively what all the variable values are, and even change them on the fly.

A general strategy is to find or create the simplest possible script that works correctly, then improve it or add stuff to it in stages. A working spreadsheet will make debugging much easier by providing lots of numbers to compare to. Also, getting the same answers in two formats (spreadsheet and code) gives a lot of reassurance that there aren’t random typo-type bugs in either format. Simplify a strangely-behaving code until it gets so simple that it works, then add the complexity back in, one step at a time, until you get that working also. You can use comment marks (#) to "comment out" lines of code temporarily.

You will be uploading your codes for automatic checking, and also for peer review. The automatic checking needs to have your code set up particular ways, to take some numbers as input, print out other numbers as output, with no extra text or on-screen plotting (matplotlib). The code checker will run your code through some paces and give what we hope will be helpful feedback on your calculations. The peer review will be to assess your coding "style", whether you have useful comments in the code, the variable names make sense, the code is logically structured: things like that. After you submit your code, you will be asked to evaluate the codes of others.

제 2 절 파이썬 라이브러리

지금부터 사용할 아이브러리와 과학계산용 파이썬 환경에 익숙하지 않은 사용자를 위해 간단히 라이브러리를 소개한다.

2.1 Numpy

NumPy(http://numpy.org)는 Numerical Python의 줄임말로, 과학계산용 파운데이션 패키지이다. 다음은 NumPy의 기능이다.

- 빠르고 효율적인 다차원 배열 객체ndarray
- 배열 원소를 다루거나 배열 간의 수학 계산을 수행하는 함수
- 디스크로부터 배열 기반의 데이터를 읽거나 쓸 수 있는 도구
- 선형대수 계산, 푸리에 변환, 난수 발생기
- 파이썬과 C, C++ 그리고 포트란 코드를 통합하는 도구

NumPy는 파이썬에 빠른 배열 처리 기능을 제공하며, 데이터 분석에서는 알고리즘에 사용할 데이터 컨테이너의 역할을 한다. 수이데이터라면 NumPy 배열은 파이선 기본 자료 구조보다 훨씬 효율적인 방법으로 데이터를 저장하고 다룰 수 있다. 또한 C나 포트란 같은 저수준 언어로 이루어진 라이브러리는 NumPy 배열에 저장된 데이터를 복사하지 않고 사용할 수 있다.

2.2 Matplotlib

https://matplotlib.org/

matplotlib 는 그래프나 2차원 데어터 시각화를 생성하는 유명한 파이썬 라이브러리다. Jhn D. Hunter 가 만들었고, 지금은 만은 배발 팀이 유지하고 있다. 출판물에 필요한 그래프를 만드는 데 맞춰졌으며 IPython에 통합되어 있어 편리하게 데이터를 살펴보고 그래프를 만들수 있다. IPython에서 matplotlib 로 생성한 그래프는 그래프 창에 있는 툴바로 특정 부분을 확대하거나 그래프의 여기저기를 인터랙트브하게 살펴볼 수 있다.

2.3 matplotlib basemap toolkit

The matplotlib basemap toolkit is a library for plotting 2D data on maps in Python. It is similar in functionality to the matlab mapping toolbox, the IDL mapping facilities, GrADS, or the Generic Mapping Tools. PyNGL and CDAT are other libraries that provide similar capabilities in Python.

Basemap does not do any plotting on it's own, but provides the facilities to transform coordinates to one of 25 different map projections (using the PROJ.4 C library). Matplotlib is then used to plot contours, images, vectors, lines or points in the transformed coordinates. Shoreline, river and political boundary datasets (from Generic Mapping Tools) are provided, along with methods for plotting them. The GEOS library is used internally to clip the coastline and polticial boundary features to the desired map projection region.

Basemap is geared toward the needs of earth scientists, particularly oceanographers and meteorologists. Jeff Whitaker originally wrote Basemap to help in his research (climate and weather forecasting), since at the time CDAT was the only other tool in python for plotting data on map projections. Over the years, the capabilities of Basemap have evolved as scientists in other disciplines (such as biology, geology and geophysics) requested and contributed new features.