And Now for Something Completely Different: Python in Cognitive Science

Mark Andrews (m.andrews@ntu.ac.uk)
Jesse Diaz (jesse.diaz@ntu.ac.uk)
Division of Psychology,
Nottingham Trent University
Nottingham
NG1 4B,UK

Keywords: Python; Programming; Scientific Computing; Numerical Computing; Computational Modelling; Experimental Design; Stimuli Presentation Software; Data Analysis;

Objectives and Scope

The objective of this tutorial is to introduce and motivate the use of the Python programming language in cognitive science research. Within the last 10 years, the development of scientific and numerical libraries in Python has grown to the point where Python can now be used as a scientific and numerical computing environment comparable to products like Matlab and Mathematica. As of yet, however, it appears that knowledge of the potential applications of Python to research in cognitive science is still rather limited. The aim of this tutorial, therefore, is to describe these areas of application and to advocate the advantages and appeals of using Python as the principal programming language in cognitive science research. Given the generality of the tools being discussed, it is hoped that this tutorial will have widespread appeal and relevance.

Outline of Tutorial

The tutorial will be divided into three main parts. The first part introduces the Python language generally. The second introduces numerical and scientific programming in Python The third part introduces how to develop computer-based psychology and psychophysics experiments using Python.

The tutorial will involve both classroom style lectures with slides and workshop style computer-based worked examples and exercises. The audience are encouraged to bring their own laptop, and all necessary software will be provided in advance.

General Introduction

In order to introduce Python, we will begin by describing the fundamentals of the Python language. We will also demonstrate how to start an interactive Python session using the ipython environment. The audience will be encouraged to follow the examples themselves using their own computers.

As part of this introduction, we will also compare Python to its alternatives, paying particular attention to comparison with Matlab. This comparison is inevitable, given that Matlab has traditionally been the principal scientific computing tool in cognitive science. Notable points of similarity between Python and Matlab are that both offer an interactive array-processing and visualization environment using high-level dynamic programming languages. Both are designed for rapid prototyping and development. Both allow for seamless extension using external modules written in compiled languages like C/C++ and Fortran. Notable advantages of Python, however, include that it is a general-purpose language whose application goes far beyond numerical array processing. Python is one of the top five programming language currently in use throughout the world. Python is a remarkably well-designed object-oriented language whose standard library is large and comprehensive. Finally, Python is non-commercial open-source software distributed according to an unrestricted software license. Likewise, its large set of third-party extension modules and libraries are, almost without exception, also distributed using unrestricted or public open-source software licenses.

Numerical and Scientific Python

The basic Python language as introduced in the previous section lacks n-dimensional numerical arrays and the ability to easily plot and visualize data. These capabilities, in addition to a large number of more special-purpose scientific libraries are provided by the Scipy/Numpy suite of modules. These libraries are seamlessly integrated with ipython to create a rich interactive array-processing and visualization environment, comparable in functionality to Matlab and Mathematica.

We will begin this section by describing ipython's capabilities more extensively than done in the previous section. These include: Interactive high-performance parallel computing for clusters and multicore architectures, an online interactive Notebook comparable to that used in Mathematica, sql-based searchable command histories, in-line graphics, and symbolic mathematics with TEX-based output.

Having established how to use ipython, the audience will be encouraged to follow the examples as we discuss the the following topics:

Arrays: General n-dimensional arrays and their operations (e.g. element-wise function application, summing, slicing, indexing, searching) are provided by numpy.

2d visualization: Plotting and visualization, especially of 2d data, are provided by matplotlib, amongst others.

3d visualization: Complex 3d graphics are provided by mayavi.

Parallel computing: Interactive high-performance and parallel programming is a built-in functionality of python.

Integration with C/C++ and Fortran: Interfaces to programs written in compiled languages like C/C++ or Fortran are pro-

vided through the use of interface generators like swig and f2py.

Computer-based Experiments

Computer-based cognitive psychology and psychophysics experiments are now almost ubiquitous in cognitive science. While these tasks have been traditionally handled by GUI-based programs like *e-prime* and *superlab*, these programs do not allow for the flexibility and control that is often demanded by researchers. While high-level languages like Matlab are being used as an alternatives to GUI-based programs, Matlab's special-purpose nature is not well suited to the non-numerical programming necessary for experimental stimuli presentation and recording. By contrast, due to the generality of its language, its extensive of widget toolkits (e.g. wxpython, pyGTK, pyQt), and video-game libraries (pyGame, pyglet), Python allows for considerable flexibility and sophistication in the design experiment software.

Currently, there are at least 4 Python-based stimulus-presentation programs: Psychopy, open-sesame, vision-egg, and pyepl. This final section will describe each in brief, but concentrate primarily on psychopy.

The aim of this section will be to discuss the principles and functionality of psychopy and then to work through examples of simple experiments (e.g. the stroop task, the lexical-decision task). Psychopy's basic object-oriented stimuli and events will be described in order to understand its extensibility. We will, however, also make extensive use of its *builder* interface that can allow from rapid development of code templates. Finally, we will discuss how to interface psychopy and Python generally with external devices such as serial response boxes that allow for precise timing of responses.

The Presenters

The main presenter for this tutorial will be Mark Andrews. Mark Andrews is a Lecturer (Assistant Professor in North American Terminology) in the Division of Psychology, Nottingham Trent University, and has a research affiliate position in the Division of Psychology and Language Sciences, University College London. His teaching primarily involves advanced statistics and experimentation methods. In this capacity, for the past two years, he has taught programming using R and Python to undergraduate and postgraduate students, with student evaluations being overwhelming costive. He has been a Python user for over 10 years, and has extensive experience with all the topics that will be covered in this tutorial. He also is very familiar with the Cognitive Science community, having presented at past conferences often, and being awarded the Computational Language Modelling prize in 2009. Jesse Diaz is a research assistant in Nottingham Trent University, with extensive experience with general Python programming and especially with the use of Python in psychology experiments, both using tools like psychopy and by using Python web-application frameworks for online experiments.

Materials

The use of Python in science is backed by a vibrant community of developers and advocates. We have been in direct contact with principal individuals in this community and they have generously offered their support, both by providing their presentation slides and other teaching materials and by providing their general advice on how to promote Python in settings such as the Cognitive Science tutorials. For example, we have been contact with Dr. Fernando Perez who is a research scientist in neuroscience at UC Berkeley. Dr. Perez is the creator and principal developer of ipython. He has kindly offered the extensive teaching materials on the ipython computing environment that are at his disposal. Likewise, we have been in contact with Dr. Jonathan Peirce who is an Associate Professor in Psychology in the University of Nottingham. Dr. Peirce is the creator and principal developer of psychopy, and has had extensive experience both teaching psychopy to students and promoting its use in psychology and cognitive neuroscience. As a result, we have a considerable body of relevant teaching materials to draw upon. Examples are available at sites like following, and elsewhere:

`http://scipy-lectures.github.com`
`http://ipython.org/presentation.html`