Python for social and experimental psychology

Alexander Pastukhov

2020-10-26

Contents

Introduction

About the seminar

This is a material for *Python for social and experimental psychology* seminar. Each chapter covers a single seminar, introducing necessary ideas and is accompanied by a notebook with exercises, which you need to complete and submit. The material assumes no foreknowledge of Python or programming from the reader. Its purpose is to gradually build up your knowledge and allow you to create more and more complex games. Yes, games! Of course, the real research is about performing experiments but there is little difference between the two. The basic ingredients are the same and, arguably, experiments are just boring games. And, be assured, if you can program a game, you certainly can program an experiment.

We will start with simple *Guess a Number* text-only game with first you and then the computer doing the guessing. Next, we will implement a classic *Hunt the Wumpus* text adventure game that will require use of more complex structures. Once we master the basics, we will up the ante by making a video game with graphics and sounds using PsychoPy library to code a classic *Memory Game*. Finally, we will create a more dynamic game by making a clone of a *Flappy Bird*.

Remember that throughout the seminar you can and should(!) always ask me whenever something is unclear, you do not understand a concept or logic behind certain code. Do not hesitate to write me in the team or (better) directly to me in the chat (in the latter case, the notifications are harder miss and we don't spam others with our conversation). As a final assignment, you will need to program a (currently mysterious) video game, which will only require the material covered by the seminar. Please inform me, If you require a grade, as then I will create a more specific description for you to have a clear understanding of how the program will be graded.

Why Python?

The ultimate goal of this seminar is to teach you how to create an experiment for psychology research. There are many ways to achieve this end. You can use drag-and-drop systems either commercial like Presentation, Experiment Builder or free like PsychoPy Bulder interface. They have a much shallower learning curve, so you can start creating and running your experiments faster. However, the simplicity of their use has the price: They are fairly limited in which stimuli you can use and how you can control the presentation schedule, feedback, etc. Typically, they allow you to extend them by programming the desired behavior but you do need to know how to program to do this. Thus, I think that while these systems, in particular PsychoPy, are great tools to quickly bang a simple experiment together, they are most useful if you understand how they create the code and how you would program it yourself. Then, you will not fill being limited by the software, as you know you can program something the default drag-and-drop won't allow, but you can always opt in, if drag-and-drop is sufficient but faster. At the end, it is about having options and creative freedom to program an experiment that will answer your research question, not an experiment that your software allows you to program.

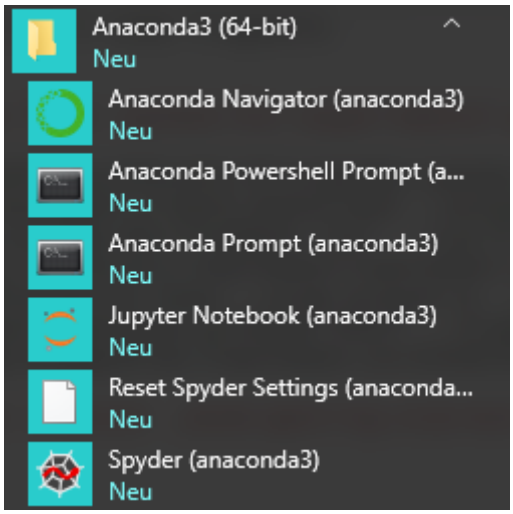
We will learn programming in Python, which is a great language that combines simple and clear syntax with power and ability to tackle almost any problem. The advantage of learning Python, as compared to say Matlab, which is commonly used in neuroscience, is that it allows you do almost anything. In this seminar, we will concentrate on desktop experiments but you can use it for online experiments (oTree), scientific programming (NumPy and SciPy), data analysis (pandas), machine learning (keras), website programming (django), computer vision (OpenCV), etc. Thus, learning Python will give you one of the most versitile programming tools that you can use for all stages of your research or work. And, Python is free, so you do not need to worry whether you or your future employer will be able to afford the license fees (a very real problem, if you use Matlab).

Getting Started

Installing Anaconda environment

First, install Anaconda, a Python distribution that includes many packages and tools out-of-the-box, makes it easy to install new packages and keep them updated. Follow this link and download the installer suitable for your platform. You can pick either 32- or 64-bit version but I would recommend the latter, so that we all have maximally similar setup (it won't really make a difference in practice, though). Follow the installer instructions and use defaults, unless you have reasons to modify them (e.g. folder location, as the drive for the default choice may have limited available space, as in my case).

After installation you will have a new *Anaconda3 (64-bit)* folder that contains links to programs.



You can use *Anaconda Navigator* that allows you to choose a specific programming environment, including Jupyter Notebook that we will use (not JupyterLab, it more versatile but we want to keep things simple at the beginning!).

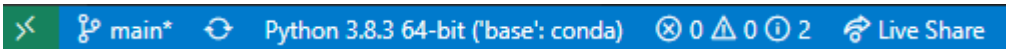
Alternatively, you can start *Jupyter Notebook* directly from the start menu. Please read the online documentation to familiarize yourself with Jupyter Notebook basic interface, e.g. how to create a new cell, run it, etc.

Installing Visual Studio Code

Visual Studio Code is a lightweight free open-source editor with strong support for Python. We will start use it in earnest, once our programs grow to be sufficiently long and complex. At the early stages, we will mostly use Jupyter notebooks and I would recommend using Jupyter notebooks using the default browser-based editor you installed as part of Anaconda. However, you can also work with Jupyter notebooks in VS Code directly.

As in case of Anaconda, download the installer for your platform and follow the instructions. Start VS Code and open any Python file, for example this one (use `Alt+click` to download it, ignore warnings, it is has only comments, so cannot harm you). When you open Python file for the first time, VS Code will suggest to install a Python extension. Do just that andinstall a linter as well when VS Code suggests that (linting highlights syntactical and stylistic problems in your code, making it easier to write consistent clear code).

Once the Python extension is activated, you will see which Python interpreter is used (you can have more than one or you may use have multiple virtual environments).



If the selected environment is the wrong one or you are simply not sure, click on it and it will open a drop-down list with all interpreters and environments you have. Consult VS Code online documentation on environments, if you need to change/add/delete environment (the exact settings may change, so looking at constantly updated online documentation is wiser than copying it here, so it would be outdated quite soon).

Installing PsychoPy

This step can wait until the first Memory Game seminar.

Download and install Standalone PsychoPy version. You can install PsychoPy as a conda package or via pip but using it as a standalone would ensure that you have all necessary additional libraries and a builder interface for the future use. We will use PsychoPy's python environment in VS Code.

Chapter 1

Guess the Number

Before we start, create a folder called *python-for-experiments* (or with some other more suitable but meaningful name) in you user folder (this is where Jupyter Notebook expect to find them). Download the exercise notebook and put it in this folder. Open Jupyter Notebook (see Getting Started, if you forgot how you do that), navigate to the folder you created and open the downloaded notebook. You will need to switch between explanations here and the exercises in the notebook, so keep them both open.