Computer practical assignment: Introduction to Python programming

Course: Energy in the Built Environment (GEO4-2522) 2021/2022

Date/Version: Aug. 1st, 2021

Abstract

This manual provides an introduction to Python programming and an overview of Python libraries and functions that will be used during the computer practical assignment in the Energy in the Build Environment (EBE) course, GEO4-2522. Through the computer practical assignment, the students are introduced to the concept of distributed generation in the built environment and demand side management for residential customers. The students will perform the computer practical assignment in groups by using Python programming. The assignment consists of two parts:

Part A: Building-integrated photovoltaics (BIPV) - Irradiance and PV performance modelling (main responsible instructor: Odysseas Tsafarakis)

Part B: Optimal Home Energy Management (HEM) - Economic optimisation of home energy management (main responsible instructor: Nico Brinkel)

After successfully completing the assignment, the students are expected to be able to apply Python programming and optimisation tools to solve a practical problem addressing performance modelling of distributed energy resources and energy management in buildings, and to report their results and findings in a clear and understandable manner.

After completing this "Introduction to Python programming" assignment, you will be able to:

- use "Spyder Anaconda" (the Python interface software),
- learn the basics of programming using Python,
- construct a program from a series of simple instructions in Python,
- learn how to deal with data (e.g., import/output data from files of different formats) and how to plot figures

Contributors and instructors

Thomas de Bruin M.Sc. (t.a.debruin@uu.nl), and Nico Brinkel M.Sc. (n.b.g.brinkel@uu.nl).

Course coordinator: dr. ir. Ioannis Lampropoulos (i.lampropoulos@uu.nl)

Recommended pre-requisites and self-study material

It is highly recommended to have some prior experience with the basic concepts of Python before you start the assignments. We recommend the Sololearn course (website: https://www.sololearn.com/learning/1073). More information about introduction in Python is provided in Section 2.2 of this manual.

Time schedule

The tutorials will take place on Tuesdays (from 11:00 to 12:45) and Thursdays (from 15:15 to 17:00) including the lectures that are specifically addressing programming techniques with Python (an introduction to Python programming, and optimisation techniques and Python tools), and an introduction to the computer practical assignments. For the detailed time-schedule please check the course guide. During the tutorials, instructions will be provided about how to complete the assignment, and students will be able to question and/or discuss any issues with the tutorial instructors.

Table of Contents

1 Introduction

1.1 Why programming is relevant for scientists working on urban energy systems?

- **Urban energy:** As over half of the human population will live in an urban environment in the near future[1], management of energy supply and demand will become essential. Developments such as transformation of the electricity grid from a centralised to a decentralised system, with distributed energy resources (DER) such as variable renewable energy sources, electric vehicles and energy storage, the transformation of the traditional energy consumers into prosumers (who both produces and consumes power), as well as potential phase-out of natural gas use and electrification of mobility will change the current urban energy landscape.
- **Big data:** The incorporation of advanced information and communication technologies (ICT) in the energy system, and the roll-out of smart metering systems contribute to the generation of huge amount of energy-related data.
- **Programming** provides automated functions to process big data on the energy sector and supports optimization models in energy systems (optimal management of DER, demand response, power balancing etc.) creating new applications and services (energy trade, yield optimisation, user comfort).

1.2 Why Python programming?

- **An open source programming language**: Python and the majority of its libraries are for free on the internet. Thus, it is accessible to anyone and the majority of the companies prefer it over other, expensive programming languages
- **Easy to understand:** Being a very high-level language, Python almost reads like English, which takes a lot of syntax-learning stress off coding beginners. Python handles a lot of complexity for you, so it is very beginner-friendly in that it allows beginners to focus on learning programming concepts and not have to worry about too much details.
- **Very popular - 5th Largest StackOverflow Community:** StackOverflow is a programming Q&A site you will no doubt become intimate with as a coding beginner. Python has 85.9k followers, with over 500k Python questions. Python questions are also the 3rd most likely to be answered when compared to other popular programming languages. Thus, most of your questions will be answered by searching in StackOverflow! In addition, Python was classified as the 2020 top programming language according to *IEEE Spectrum*'s fifth annual interactive ranking of the top programming languages.

[1] Global population is expected to surpass 9 billion by 2050, and the United Nations projects that by 2050, 6 billion people will live in cities.

1.3 Myworkplace environment – Python

As a student, you go online anywhere and all the time. A flexible mix of online and face-to-face learning fits in well with your lifestyle. Making your own choices is part of that. And that is what you can do at Utrecht University. Thanks to MyWorkplace, Utrecht University's digital work and study space, from now on you will be able to use the software for your studies in a very flexible way. And you'll be able to do that when and where you like using whatever device you want.

In order to use MyWorkplace, you need the tool or App called ´Citrix Receiver´ that can be downloaded for free. Make sure you install this Citrix receiver first before logging on to MyWorkplace the first time. Without the Citrix Receiver, a light version of MyWorkplace will be started, and you'll miss important functions within MyWorkplace.

To start MyWorkplace after installation of the Citrix receiver, open Citrix receiver and connect with your e-mailadres (initials.surname@students.uu.nl) and your Solis password. At the "Apps" surface (top right of Citrix receiver) find and run "Spyder Anaconda"

1.4 Working with Spyder

Once you have logged-in to MyWorkplace through the Citrix you can select to open Spyder Anaconda3. Spyder (Scientific PYthon Development EnviRonment) Anaconda3 is a very useful Python platform. It is preferred from original python interface (IDLE) because it is user friendly and provides extra features to the user specifically for scientific applications. In this section, some of these features are presented. The familiarisation of the user with these features can save significant programming time and improve the quality of work.

1.5 Spyder interface.

When you open Spyder you will see different panels. In the status bar, you can choose the panels that you want to see through: View -> Panels. The panels that are suggested to be used are the following:

- Editor
- IPython Console
- Variable Explorer
- Help
- Outline

Editor

Editor is the area where you can write, edit and save your code. Editor is the main working environment. By pressing the green play button on the Tool bar (or F5) you can run the code (and save it at the same time). In case that you want to run small parts of your code (e.g. one or more lines) you can mark the code that you want to run and press Ctrl+enter (or cmd + enter for MAC).

IPython Console or Python console

IPython Console is the Panel where Python is running and where you are executing the code (with the steps described above). You can also write and run the code directly but it will not be saved after you close the program.

Tip: If you are writing in IPython console (or in the Editor), after the name of a library or function of the library add a dot "." and press Tab. (e.g. 'pandas.' and press 'tab'), then all the available functions will appear.

Variable explorer

Variable explorer is the Panel where all the variables which have been set in the selected IPython Console are presented, accompanied with some information. If the variable is of pandas DataFrame or Series, by clicking on the variable name, a new window is opening and the DataFrame is presented (in a form like excel sheet). If you cannot find a specific variable, check the three small stripes at the top right corner. It might be the case that certain variables are excluded, and these settings can be adjusted here.

Help

In the Help Panel, information about functions is presented. On the upper left side of the panel you can choose your source which means that the displayed info will be from functions written on IPython console or on Editor. Then, every time you write a function, its information will be displayed on Help. If they are not displayed, place the cursor on the function and press ctrl+I (or cmd+I on Mac). The important information for every function is its parameters, that you have to enter in order to use the function properly. That makes the panel Help very important for writing a code.

Outline

It is basically the table of contents of your Editor. It displays your created functions and also your notes (if they begin with 3 hashtags '### note')

2 Prior to the practical assignment

Prior to the practical assignment it is highly recommended, especially for the students that are not familiar with Python programming, to complete the following steps:

2.1 Complete the online course until Sep. 10, 2021.

It is highly recommended to all enrolled students to the Sololearn course (https://www.sololearn.com/learning/1073) before the beginning of the EBE course. Please start as early as possible with the online course (if possible then even prior to the start of Period 1).

Online Sololern course:
A helpful tool to get acquainted with Python is the free Sololearn course (website: https://www.sololearn.com/learning/1073), which can also be accessed through an Android and iOS application. You do need to create an account, but further use is for free. Some useful info and advices:

- It is highly recommended to start as early as possible with the online course (if possible then even prior to the start of Period 1).
- The Sololearn course is similar to DuoLingo, there are several modules with small lessons and a quiz at the end which advances you to the next module.
- Each exercise takes about 1-3 minutes.
- There are also 'practice' exercises available for additional learning. First, it is advised to do the lessons, and do the practice lessons only if you have time to spare or really enjoy it.
- For the EBE course it is advised to finish at least module 5: More types. Which will probably take 3-5 hours.
- Due to the lesson-based structure it is advised to make a habit of finishing a lesson whenever you have some free time.
- Good luck, and keep in mind that there is also a substantial forum integrated with the lessons, but if you get stuck, feel free to ask for advice.

By following the above-mentioned tips, you will be equipped with the necessary resources for getting familiar with the basics of Python programming. More in-class support will be provided during the tutorials.

It is allowed to follow the course during the first week of the EBE course but for a maximum learning output, it is highly recommended to complete it before starting the EBE course.

2.2 Follow the introductory lecture 'Introduction to Python programming'

During this lecture, it will be expected that students will get prepared to ask any question regarding the above. The lecture will be in the form of a masterclass, where an example script in Python will be presented. The example Python script will consist of examples of functions and combinations of functions from the libraries *pandas*[2] (reading and processing data), *matplotlib*[3] (visualising data) and *pvlib*[4] (irradiance and PV performance models), and *numpy*[5] (multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays). The provided examples will be similar to the data analysis that will be used during the two main parts of the practical assignment. Thus, you can return to this file to get support regarding any problems that you might face while processing the data for the practical assignment.

2.3 Python tutorial:

This tutorial serves as a supporting learning material to the online course. It reviews some programming concepts introduced in the online course. In addition, it covers some programming modules and tools that are missing in the online course and needed for the main practical assignments. The tutorial is structured as follows:

- Defining Python functions.
- Using pandas (reading CSV files, showing/checking data, selecting specific data, using the index to select data, resampling data and plotting data).

2.3.1 Defining Python functions

If you have to perform a complex action, or even a simple action, in Python multiple times, it can be convenient to define a function. These functions can process a set of inputs, for instance in the first assignment you are asked to calculate the RMSE of a model compared to measurements. So, here we define the function SUM(modeled_values, measured_values):

```
def SUM(modeled_values, measured_values):
    sum = modeled_values + measured_values
    return sum
```

The value(s) that you want the function to return follow the command **return**

2.3.2 Introducing and using pandas

Reading CSV files

The most convenient way of reading CSV files into python is to use the package **pandas**, which has built-in functionality for parsing these files. Import this into Python using:

[2] Link: http://pandas.pydata.org/pandas-docs/stable/
[3] Link: https://matplotlib.org/
[4] Link: http://pvlib-python.readthedocs.io/en/latest/package_overview.html (especially the API reference section is helpful for figuring out how to use specific functions of the package)
[5] Link: http://www.numpy.org/

```
import pandas as pd
```

Read a CSV file with:

```
data = pd.read_csv(path_to_file)
```

The path_to_file parameter is the full file path including the filename. On Windows, use double slashes "\\" in the file-path, or a single forward slash "/"! . For example: 'C:\User\\Courses\\EBE\\file.csv'. If it is not working type an r in front of the path like r'C:\User\\Courses\\EBE\\file.csv'

Because a CSV file is text only, we need to add a few arguments to this function. For instance, the CSV files provided here use semicolons (";") to separate the data columns. Furthermore, in pandas, it is convenient to use a so-called index column, in this case a timestamp (date and time). For the data from KNMI and UPOT, we can read the data, and parse the dates (otherwise they will be loaded as text strings) with:

```
KNMI_data = pd.read_csv(filepath, sep=";", index_col="dates", parse_dates=True)
UPOT_data = pd.read_csv(filepath, sep=",", index_col="timestamp", parse_dates=True)
```

The index_col parameter refers to the names of the columns in the file. So, you need to check these first before reading the file into python.

Pandas is a very comprehensive package for loading, processing and analysing data in a structured manner. It has countless functions described in the documentation (http://pandas.pydata.org/pandas-docs/stable/). Below we highlight a few of them.

Showing/checking data

If you want to view your data (e.g. to check if importing data worked) you can do it through the panel: "Variable Explorer". In this panel, the list of set variables is appeared. By double clicking on the name of the dataframe, a new window appears, where the dataframe is presented in a form similar to an excel sheet.

You can also see the head (first rows) or tail (last rows) of the loaded data on the IPython Console by:

```
data.head()

data.tail()
```

Where data is the pandas "dataframe" (like the one you have made when reading a CSV file, see above).

Indexing

There are different ways of indexing in Pandas. Some examples are presented at the given script ‘Python _EBE_example.py’, under the ###INDEXING IN PANDAS.

It is recommended not to run the whole script but run its line separately and to observe on IPython console or at variable explorer the changes and the new data-frames or series that you are creating through the indexing process.

Using the index to select data

If we want to select parts of the data, the first way would be to use indexing. In the example above, where we read data from a CSV file and set the timestamp as the index column, we could use the date/time to select a short time-range:

```
start_date = “2015-06-01 15:00”

end_date = “2015-06-07 15:00”

short_range = KNMI_data.loc[start_date:end_date]
```

Resampling data

To resample data, we use the pandas function resample. We call this function on the declared dataframe, e.g. a dataframe called UPOT_data, read from CSV like above. In this example, the data is binned in 600 second intervals and these data are averaged:

```
resampled_UPOT_data = UPOT_data.resample(‘600S’, how = ‘mean’)
```

For all possible resampling time frames and methods (e.g. other than taking the mean of the bins), look at the pandas documentation:

(http://pandas.pydata.org/pandas-docs/stable/timeseries.html#resampling)

Plotting data

To plot our data, we use the **matplotlib.pylab** (or equivalent, **matplotlib.pyplot**) module, import this into Python using:

```
import matplotlib.pylab as plt
```

With matplotlib you can make an enormous variety of plots. E.g. scatter plots (**ax.scatter()**), barcharts (**ax.bar()**), boxplots (**ax.boxplot()**) and many, many other types of plots. Please check the documentation on https://matplotlib.org/gallery/index.html#pyplot-examples for a lot of examples and more information. Furthermore, some basic examples are presented at the script ‘Python _EBE_example.py’ under the ### DATA VISUALIZATION.

You can save files using the function below. Indicate a filepath, including the filename and extension. We suggest exporting to PNG with a resolution of 400 dpi:

```
plt.savefig(filepath, resolution=400)
```

Data analysis with Pandas, Numpy and SKlearn

Most of the statistical tools that will be used for data analysis through the exercise (e.g. Mean Square Error, mean value, R2) are already included as functions, either in **pandas** or in the libraries **numpy** and **sklearn**. Thus, it is recommended to use google to find out these functions instead of creating your own.

Numpy and sklearn are libraries which includes different mathematical tools and imported as:

```
import numpy as np

import sklearn
```

Some basic examples are presented at the script 'Python _EBE_example.py' under the:
DATA ANALYSIS AND SIMPLE MATHEMATICAL FUNCTIONS ON PANDAS.

2.3.3 Python dictionaries

In assignment A, you will see that you will sometimes need to repeat the same steps for different variables, for instance if you have to repeat a certain analysis for different façades. One way to do this, is that you copy a script but use different variables. However, this is time consuming and prone to errors. A more convenient way to do this by using dictionaries, in which you can store data using 'for-loops'.

An example, if you have the following dataframe called 'shoesizedf', and need to determine the square of shoesizes for Ioannis, Odysseas and Nico.

Name:	**Shoesize:**
Ioannis	43
Thomas	46
Nico	46

One way to solve this problem is creating separate variables for all three names, which would make the code as follows:

```
ioannisshoesize=shoesizedf.at['Ioannis','Shoesize']

ioannisshoesizesquared= ioannisshoesize**2



thomasshoesize =shoesizedf.at['Thomas','Shoesize']

thomasshoesizesquared= thomasshoesize**2
```

```
nicoshoesize =shoesizedf.at['Nico','Shoesize']

odysseasshoesizesquared= nicoshoesize**2
```

Already for this very simple exercise, you need a large number of variables. Especially when you have to do complex calculations with a large number of variables, this is inconvenient. A much simpler way is to use dictionaries, which allow you to automatically create variables using for-loops:

```
shoesizedict={} #create the dictionary

for name in ['Ioannis','Odysseas','Nico']: #for-loop over all names

    shoesize= shoesizedf.at[name,'Shoesize'] #automatically get value from dataframe

    shoesizedict[name]= shoesize**2 #make a new key in dictionary, and save value
```

As you can see, using dictionaries is more efficient and less prone to errors.

3 Appendix

Even though you will program through the myworkplace environment, herewith are some instructions about how to install Python and libraries without an UU account.

3.1. Installation of Python and libraries

To install python, it is most convenient to make use of a so-called Python distribution. We suggest you use *Anaconda*, as it includes a lot of python packages that will be needed for the practical assignment, as well as a package installer. You can download Anaconda for Windows, Mac and Linux by following this link. We assume you use the Python 3.8 version. Once the installation of the Anaconda is complete, open the *Anaconda prompt*. Here, you can install packages as follows:

```
conda install x
```

Where x is the package name. If this does not work, it means that either the package is not in the *Anaconda* channel, or there is a version mismatch.

In this case you can add a channel with:

```
conda config --add channels x
```

Where x is a link to the channel url. Another option is to try to use the command pip:

```
pip install x
```

3.2. Install Gurobi optimiser

3.2.1. Installation for Mac OS

Gurobi supports Python 2.7 and 3.6 for Mac OS.

From an Anaconda prompt terminal, run the following command to add the Gurobi channel to your default search list:

```
conda config --add channels http://conda.anaconda.org/gurobi
```

Install the Gurobi package using the command:

```
conda install gurobi
```

You can remove the Gurobi package at any time by issuing the command:

```
conda remove gurobi
```

Install a Gurobi license

You are now ready to use Gurobi from within Anaconda. Your next step is to launch either the Spyder IDE or Jupyter Notebook.

3.2.2. Installation for MS Windows

Gurobi supports Python 2.7 and 3.6 for Windows.

From an Anaconda terminal, run the following command to add the Gurobi channel to your default search list:

```
conda config --add channels http://conda.anaconda.org/gurobi
```

Install the Gurobi package using the command:

```
conda install gurobi
```

You can remove the Gurobi package at any time by issuing the command:

```
conda remove gurobi
```

Install a Gurobi license

You are now ready to use Gurobi from within Anaconda. Your next step is to launch either the Spyder IDE or Jupyter Notebook.