# Automation Guide for Non-Programmers

## Introduction to Scripting

A script is used to quickly accomplish a repetitive task a multitude of times, for example, changing the name of many files that follow one naming convention, or drawing a shape not normally found in AutoCAD that has special representation within the company. Both these things have already been done to some degree by previous co-ops for Plan Group. In the case of company policy change or other unforeseeable events, scripts may need to be edited or rewritten to comply. This guide hopes to set a baseline level of scripting knowledge so that if need be, any employee can alter previously existing scripts to their needs while keeping to general programming conventions as to prevent a messy and fragile code base. Rather than teach python syntax from the start, the guide will feature sample code and explain line by line as a way to fast track script writing. There are many good resources online to further delve into if you wish to read more on python as a language.

Python has been installed on this computer. Python is constantly being updated but as of writing this, the version of Python installed is 3.7.3. If you find this version of Python to be obsolete, you can go to [python.org/downloads/](https://www.python.org/downloads/) to download the latest version of Python. Pycharm and PowerShell will have been installed too. PyCharm is a python specific IDE (integrated development environment) made by JetBrains. It will require a license but JetBrains offers free licenses to university students. If you need to install it, go to [jetbrains.com/pycharm/download/](https://www.jetbrains.com/pycharm/download/). If PyCharm is unavailable, there are many other IDEs. Visual Studio is a strong platform built by Microsoft[[1]](#footnote-1) that’s free.

## Updating Dates and Revisions

The simplest and most common task that can be automated is updating the date and revision of PDFs. Plan Group has a defined naming convention that will look something like[[2]](#footnote-2):

“PGE-09-13-01-O-U - PUBLIC ADDRESS - LEVEL 09 AHU 13-01 (PG1 MAR 10, 2020)”

Previous code has been written to automate naming these when the sheets are being created in Revit. It can be tedious to fix any mistakes aside from reprinting all the PDFs directly from Revit.

The following code has been written in order to tackle this. The next few pages will explain how the code works, and how it runs. For how to use this code, skip until page [x].

import os  
  
# Change revision, date, and directory, and run the program, leave everything else  
# Function to rename multiple files  
def main():  
  
 # Copy the location of the files in the quotations. Make sure to include double backspaces.  
 # E.g. "C:\\Users\\pass.ecs.coop\\Desktop"  
 directory = "C:\\Users\\pass.ecs.coop\\Desktop\\Test Files"  
  
 # Revision number: if revision is PG2, set revision = "2"  
 revision = "1"  
  
 # Date: set to desired date format within quotations  
 date = "MAR 10, 2020"  
  
 os.chdir(directory)  
   
 # Loop through directory  
 for filename in os.listdir():  
  
 # Finds length of file name, removes ending (PG# DATE) to be replaced  
 for i in range(0, len(filename)):  
 if filename[i] == "(":  
 break  
  
 # Renames files  
 src = filename  
 dst = filename[:i] + "(PG" + revision + " " + date + ").pdf"  
 os.rename(src, dst)  
  
  
# Driver Code  
if \_\_name\_\_ == '\_\_main\_\_':  
 # Calling main() function  
 main()

To go through the code, the first thing to notice is:

import os

What this line does is allow use of pre-programmed methods from the OS module that interact with the computer’s operating system. For example, “os.chdir” is a method that couldn’t be used if import os wasn’t included. You can simply include import os at the top of your code and forget about it.

The next important thing to notice are the comments.

# Change revision, date, and directory, and run the program, leave everything else

These are any lines preceded by a “#” and are greyed out. They don’t effect the code in any capacity and are there solely for the programmer and any readers. Comments are a vital part of any code as it helps both the author and future readers to understand the thought process behind chunks of code. Code is often done in a linear progressive fashion, so understanding the idea behind one bit of code, will help with the next, and so on. You can even comment out chunks of code that don’t work as a way to skip over the code without deleting it.

Another line that can mostly be ignored is:

def main():

The importance of this is that it encompasses and defines the entire program. You can treat the entire program as a function named “main”. Later, the driver code will be explained which will include this part. For now, the only thing to keep in mind is that Python groups your code through colons and tabs. After this colon, any further code indexed one or more to the right is “under” the function “main”.

The next few lines are variables that require being changed by the user.

directory = "C:\\Users\\pass.ecs.coop\\Desktop\\Test Files"  
revision = "2"  
date = "MAR 10, 2020"

In programming variables are stored information that is used throughout the code, but can be changed without having to rewrite all the many lines that use the information. In the example code:

name = "John"  
print("Hi " + name)  
print ("It's nice to meet you " + name)  
print ("How have you been " + name)

The outputs of all 3 print lines can be changed just by changing the changing the name variable. In the script, the variables are only used once, and thus aren’t entirely necessary, but keeping them as variables keeps the code neat and keeps the code flexible and easy to understand. As the instructions say, to change the variables, simply change the content within the quotations.

The next two lines are for searching through the computer’s files, and contain methods imported from the import os line discussed above. That’s the reason for os period notation.

os.chdir(directory)  
for filename in os.listdir():

This is also the first instance of one of the variables mentioned above being called as a parameter. Placing the directory variable in the brackets as the parameter to the method is the same as typing out os.chdir("C:\\Users\\pass.ecs.coop\\Desktop\\Test Files"). Having the variable keeps the code cleaner, and separates the idea of the information from that of method. The second method os.listdir does not require a parameter, and is used in the for-loop. What a for-loop does is run a chunk of code a certain number of times. The method creates a list of names within a directory and the for-loop can be understood as “for each file in the list, iterate the code once”. This is an iterator-based for-loop, the code within the loop will be executed once per PDF. “Filename” is another created variable which stores the names of the PDFs. Each time the for-loop iterates, the variable filename is overwritten with new information, that being the next PDF’s name. Any further use of the variable in the loop will be done with that new file name.

# Finds length of file name, removes ending (PG# DATE) to be replaced  
i = 0  
for i in range(0, len(filename)):  
 if filename[i] == "(":  
 break

This nested for-loop is used for

1. There may be a Visual Studio installation guide at the end of this. If not, it’d be a good idea to add. [↑](#footnote-ref-1)
2. As of writing this, I’m not sure if there exists code to change pdf names directly. The network is down so I don’t have access to many of the company’s files. Regardless, the objective of this guide is so that the function can be easily edited by any employee. [↑](#footnote-ref-2)