

# Python

## Module 5

*Dictionaries and Extending Hangman*





# Content

This Module .....	2
Recap of Module 4 .....	2
Answers Exercise Module 4.....	3
Exercise.....	3
Solution.....	3
Dictionaries.....	6
Extending the Hangman Game .....	6
Preservation tools on the Command-Line Interface.....	7
Exercise Preservation Tools in the Command-Line Interface .....	7



## This Module

### For this module, make sure you:

- Recap what you have learned in the last module. Check your own notes, or read the summaries provided at the end of the chapters of Al Sweigart books.
- Read the guide and make sure you understand everything.
- Read chapter [5](#) of *Automate the Boring Stuff*.
- Read and work through chapter [9](#) of *Invent Your Own Computer Games With Python* while following the guide. Be able to understand the concepts covered in those chapters.

### Further learning:

- If you want to read more about dictionaries and practice with them, follow the [Python Dictionaries](#) chapter on W3Schools. Via multiple assignments you can try out the basics.
- If you want another view on lists and dictionaries, you can read chapter 5 of Michael Dawsons, [Python Programming for the Absolute Beginner](#), that covers the same topics. It describes another way to create the Hangman Game.

## Recap of Module 4

Before creating Hangman, you have experienced that it can be beneficial to make a flowchart. This way, you can make changes to your program more easily and identify problems by thinking about how the code works. In order to create the Hangman Game, you have used ASCII art. Furthermore, a new data type has been introduced: **Lists**. A list has values that can contain other values. You can recognise a list in your code by the square brackets, [ ].

You can use methods while using lists. **Methods** are functions attached to a value. Two methods from module 4 are repeated here: You made up words that can be guessed in the game (in a list) and typed all those words only separated with a space. The `split()` method evaluates to a list with each word in the string as a single list item. The `append()` method adds an argument to the end of the list.



## Answers Exercise Module 4

### Exercise

For this exercise, you have created a Python script that will help you check several things and saved the output in a CSV file.

### Solution

```
import pandas as pd
import os

# === START USER SPECIFIC CHANGES ===
metadata_path = "[PUT PATH HERE]"
fileName = "[FILENAME HERE]"
# === END OF CHANGES ===

# Import Excel file
name_file = os.path.join(metadata_path, fileName + ".xlsx")
file = pd.read_excel(name_file)

# Extension check (final 4 characters have to be .pdf)
file['extension'] = file['File'].str[-4:]
index_wrong_extensions = file.index[file['extension'] != '.pdf'].tolist()

if index_wrong_extensions:
    print("Wrong file extensions!!!")
    print(index_wrong_extensions)
# Remove aiding column
file.drop(columns=['extension'], inplace=True)

# Create new file names (spaces to underscores)
name_file_clean = name_file.replace(" ", "_")
file['file name'] = name_file_clean

# Columns for deduplication
col_address = [21, 22, 23, 24, 25] # 0-based index
col_names = [17, 18, 19, 20]
col_pdf = [0, 29, 30, 31]
col_bag_building = 26
col_bag_object = 27

# Check unique dossiers
dossiers = file['Registrationnumber(s)'].dropna().unique()

for dossier in dossiers:
    idx = file[file.iloc[:, 1] == dossier].index
```



```
# Double PDF
dup_pdf = file.loc[idx, file.columns[col_pdf]].duplicated()
file.loc[idx[dup_pdf], file.columns[col_pdf]] = pd.NA

# Double BAG building
dup_building = file.loc[idx, file.columns[col_bag_building]].duplicated()
file.loc[idx[dup_building], file.columns[col_bag_building]] = pd.NA

# Double object
dup_object = file.loc[idx, file.columns[col_bag_object]].duplicated()
file.loc[idx[dup_object], file.columns[col_bag_object]] = pd.NA

# Double names
dup_names = file.loc[idx, file.columns[col_names]].duplicated()
file.loc[idx[dup_names], file.columns[col_names]] = pd.NA

# Double addresses
dup_address = file.loc[idx, file.columns[col_address]].duplicated()
file.loc[idx[dup_address], file.columns[col_address]] = pd.NA

# Save as CSV
output_path = os.path.join(metadata_path, fileName + " after data prep.csv")
file.to_csv(output_path, index=False, na_rep="")
```

1. First the needed libraries are imported.
2. Then the path to the location where the Excel file is situated has to be defined, using `metadata_path`. The location in this solution is an example. Make sure to use forward slashes, not backslashes.
3. Define the Excel file you want to analyse and deduplicate by using `fileName`. Note that the filename has no extension.
4. Import the Excel file by first joining the `metadata_path`, `fileName` and extension of the Excel file and then use the `pd.read_excel(name_file)` function to read the Excel. These are Pandas functions.
5. The extension check checks in the Excel file the last four characters of the filenames in the first column. The code in the solution converts the filenames in the Excel file to string and then looks at the last four characters `[-4]`. The next line of code creates an index for every file that doesn't have `.pdf` at the end, using the not equal operator and the `.tolist` function.
6. Use the `if` function to print the list of wrong file extensions, created in the previous lines of code.
7. The list function creates an extra column. Remove this column using the `file.drop` function. This is a specific Pandas function.
8. Create new file names using the `.replace` function and make sure the new names are part of your updated Excel file.
9. Connect the columns per item that you want to deduplicate, so that the script knows in which columns to look for double data. In this case, address, names and pdf need several columns combined to see if the metadata needs to be deduplicates, the BAG building and BAG object columns only need one.



10. Use the Pandas `.dropna().unique()` function to ignore all unique items in the Excel sheet.
11. Use the 'for' 'in' function to deduplicate the five items using the Pandas functions for cleaning wrong data (`file.loc[idx, file.columns[col_pdf]]`) and the deduplication function `.duplicated()`. Use the `.pd.NA` Pandas function to provide a "missing" indicator, should anything go wrong in the previous lines of code.
12. Save the new Excel file by joining the `metadata_path`, `fileName` and a new unique identifying piece of text plus the extension, then save the file by using the Pandas function `file.to_csv`.

There are three files in the original Excel spreadsheet with extensions different from `.pdf`. After running the Python script, your IDLE or Mu shell will show the list created with the Extension check function and show you the rows where the exceptions are.

```
= RESTART: C:/Users/EvavandenHurk-van'tK/AppData/Local/Programs/Python/Python313
/BBdeduplication.py
Wrong file extensions!!!
[0, 144, 372]
```



## Dictionaries

We have seen that lists in Python are a very helpful way to handle large amounts of data. In this module, you will learn that combining lists with dictionaries is a flexible way to not only handle the data, but also access and organise it. We will return to the Hangman game in this module, and improve the code with the help of dictionaries.

Read and work through [Chapter 5](#) of *Automate the Boring Stuff*.

## Extending the Hangman Game

Now it is time to get back to the Hangman Game and make use of dictionaries next to lists. Chapter 9 is dedicated to extending the Hangman game in *Invent Your Own Computer Games with Python*. In this chapter, you will not only use the data dictionary data type, but also create the ability to change the game's difficulty level by reducing the number of guesses a player has.

Copy your code from the last module, and save it as a new file (for example `Hangman_dictionary.py`). This way, you can review both versions of the game later on.

Read and work through [Chapter 9](#) of *Automate the Boring Stuff*.

Optional challenge: make the Hangman Game more digital preservation themed.



## Preservation tools on the Command-Line Interface

For the expert session in October, we are joined by Lode Scheers and Nastasia Vanderperren. They will show us how to use preservation tools on the Command-Line Interface (CLI), specifically ExifTool. If you want to follow along live during the expert session, make sure to follow the steps below ('To install before starting'), beforehand.

### Exercise Preservation Tools in the Command-Line Interface

If you are still new at using the command-line interface (CLI), please check the [Command-Line Interface Guide](#).

Please note that these practice based exercises are focused on trying out tools via the CLI. We do not provide answers for the exercise in the next module. Let us or the experts know if you are stuck or need help.

#### To install before starting<sup>1</sup>

Windows users need to install [Scoop](#)

1. open Windows PowerShell
2. paste `Set-ExecutionPolicy RemoteSigned -Scope CurrentUser`  
`Invoke-RestMethod -Uri https://get.scoop.sh | Invoke-Expression` in the terminal and press <enter>
3. ready!

macOS users need to install [Homebrew](#)

1. open your terminal
2. paste `/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"` in your terminal and press <enter>
3. in case step 2 ends with an error, paste `xcode-select --install` in your terminal and press <enter> and repeat step 2
4. ready!

#### Editing Image Metadata with ExifTool

In this exercise, you will:

1. Download an image (e.g. from [Wikimedia Commons](#)).
2. Install command line programs by using a package manager
3. Inspect its embedded metadata (information about the image, like creator, description, etc.).
4. Modify the metadata to include your own values for Creator and Description.

---

<sup>1</sup> Struggling with installation? You can ask your question in the Discord server or you can contact Lode ([lode.scheers@meemoo.be](mailto:lode.scheers@meemoo.be)) or Nastasia ([nastasia.vanderperren@meemoo.be](mailto:nastasia.vanderperren@meemoo.be)) directly.



5. Save the modified metadata back into the image.

This is useful if you want to add descriptive information - or remove privacy information - to images you post online, or if you want to practice automating workflows that involve images.

### What do you need?

- A terminal (Command Prompt, PowerShell, or Mac/Linux terminal).
- A package manager for installing/updating/removing command line programs (optional, but recommended)
  - [Scoop](#) for Windows
  - [Homebrew](#) for macOS
- At least one downloaded image (e.g., .jpg file).

Optional (extra challenge): Python users can try to do this with Python as well.

This assignment has four parts: installing ExifTool, inspecting your metadata, exporting your metadata to a CSV file, and re-importing your modified metadata.

### Part 1. Installing ExifTool

[ExifTool](#) is a command-line tool for reading, writing and editing metadata in images.

1. Open your command-line interface (command prompt or terminal)
2. Install ExifTool by using your package manager (scoop or brew)

#### *Windows*

```
scoop install exiftool
```

#### *macOS*

```
brew install exiftool
```

If you don't want to use or install a package manager, you can [download ExifTool](#) (it works on Windows, Mac, and Linux).

### Part 2. Inspecting your metadata

1. Make sure you have downloaded one or more images and have installed ExifTool
2. Make sure your command-line interface (command prompt, powershell or terminal) is opened
3. Navigate to the folder where your images are stored using the `cd` command
4. Run ExifTool on an image. This will list all the metadata currently inside the image.

```
exiftool [yourimagename.jpg]
```

### Part 3. Exporting your metadata to a CSV file

1. Use ExifTool to export metadata from all images in a folder:

```
exiftool -csv *.jpg > metadata.csv
```



2. Open metadata.csv in a spreadsheet program (e.g, Excel or Libreoffice)
3. Check if the Creator and Description columns exist
  - a) If they exist → try to change some values
  - b) If they don't exist → add those columns and fill in values

#### **Part 4. Re-importing your modified metadata**

1. Save your edited metadata.csv
2. Import it back into your images:

```
exiftool -csv=metadata.csv -overwrite_original_in_place *.jpg
```

This updates the metadata in each image according to the CSV file.