Python

Module 4

Lists and the Hangman Game

Bits and Bots
study group

Content

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 4 of Automate the Boring Stuff.
- Read and work through chapters 7 and 8 of Invent Your Own Computer Games With Python while following the guide. Be able to understand the concepts covered in those chapters.

Further learning (optional):

- If you want to practice more with lists, check out: Python Tutorial (w3schools.com). Via multiple assignments you can try out the basics.
- Chapter 5 of Michael Dawsons, *Python Programming for the Absolute Beginner*, describes another way to create Hangman. Please do note that this chapter covers dictionaries as well, something that we will cover in Module 5.

Recap of Module 3

In the previous module, you have been working with new functions and loops. The `while` loop will only run if a certain condition is `true`. With the `sleep()` function you are able to build up the suspense in a game, as we have seen in Dragon Realm. Furthermore, you have learned how to create your own function. A function is a small program within your code, that only runs when the function is called. By using functions you create small pieces of code to structure your program into sections. If you understand how functions work and how to use them, you can save yourself some time, because you can reuse functions. Also, it is easier to fix bugs.

Using a debugger is great to understand what your program does, because you will have a closer look at your code line by line. The `logging` module facilitates this when your code is running. Furthermore, we have seen three types of bugs:

- Syntax errors: Typos in the code like forgetting a quotation mark or a bracket.
- Runtime errors: Bugs that happen while the program is running. The traceback gives an error message showing the line containing the problem
- Semantic errors: These bugs don't crash the program, but prevent it from doing what the programmer intended it to do. The program could crash later on, but it won't be immediately obvious where the semantic bug happened.

Answers Exercise Command-Line Interface

Windows

```
@echo off
cd /d "C:\Path\To\Your\TargetFolder"
echo Navigated to %cd%

echo Listing contents of %cd%:
dir

set "NEWFOLDER=CopiedFiles"
mkdir "%NEWFOLDER%"
echo Created folder: %NEWFOLDER%

set "ORIGINAL=example.txt"
set "COPY=%NEWFOLDER%\example_copy.txt"

if exist "%ORIGINAL%" (
    copy "%ORIGINAL%" "%COPY%"
    echo Copied "%ORIGINAL%" to "%COPY%"
) else (
    echo ERROR: File "%ORIGINAL%" not found.
    exit /b 1
)

echo Calculating SHA512 checksum for %COPY%:
certutil -hashfile "%COPY%" SHA512
```

Notes to Windows script

- By using echo commands throughout your script, you make sure that you've used the right command. For example, `echo Navigated to %cd%` would print as `Navigated to C:\Path\To\Your\TargetFolder` on your screen.
- Replace `C:\Path\To\Your\TargetFolder` with the actual folder path.
- Replace `example.txt` with the file name you want to duplicate.
- If you prefer MD5, change SHA512 to MD5 in the certutil command.
- Save the batch script as a `.bat` file. Open the cmd prompt (search for cmd in the search bar and press `ENTER`. Navigate to the right folder where the batch script is saved, and run the file by typing the file name, such as: `C:\Users\Bits_and_Bots\Python_Scripts> BatchScript.bat`

Mac

```
#!/bin/bash

cd "/Users/yourusername/TargetFolder"
echo "Navigated to $(pwd)"

echo "Contents of $(pwd):"
ls -la

NEWFOLDER="CopiedFiles"
mkdir -p "$NEWFOLDER"
echo "Created folder: $NEWFOLDER"

ORIGINAL="example.txt"
COPY="$NEWFOLDER/example_copy.txt"

if [ -f "$ORIGINAL" ]; then
    cp "$ORIGINAL" "$COPY"
    echo "Copied $ORIGINAL to $COPY"
else
    echo "ERROR: $ORIGINAL does not exist"
    exit 1
fi

echo "SHA512 checksum of $COPY:"
shasum -a 512 "$COPY"
```

Notes to Mac script

- The script starts with `#!` This is called the shebang marker. Without it, the script may run in a different shell or fail to run entirely in some environments. You can also use `#!/usr/bin/python3` on the first line for Python scripts.
- Replace `yourusername` and folder paths with your actual username and target directory.
- In this example, we included an if statement. It starts with if and ends with fi (literally 'if' spelled backwards). The batch script checks if the said file exists, in that case the condition in the statement is true, so the file will be copied. If the file does not exist, this script will print '`ERROR: example.txt does not exist`' on your screen.
- If you want to calculate a MD5 checksum, use: `md5 "$COPY"`
- Save the batch script as a `.sh` file. To run the file on the macOS Terminal, open Terminal, navigate to you folder and call the script: `chmod +x /path/to/script.sh && /path/to/script.sh`

Linux

```
#!/bin/bash

cd "/home/yourusername/TargetFolder"
echo "Navigated to $(pwd)"

echo "Contents of $(pwd):"
ls -la

NEWFOLDER="CopiedFiles"
mkdir -p "$NEWFOLDER"
echo "Created folder: $NEWFOLDER"

ORIGINAL="example.txt"
COPY="$NEWFOLDER/example_copy.txt"

if [ -f "$ORIGINAL" ]; then
    cp "$ORIGINAL" "$COPY"
    echo "Copied $ORIGINAL to $COPY"
else
    echo "ERROR: $ORIGINAL does not exist"
    exit 1
fi

echo "SHA512 checksum of $COPY:"
sha512sum "$COPY"
```

Notes to Linux script

- The script starts with `#!` This is called the shebang marker. Without it, the script may run in a different shell or fail to run entirely in some environments. You can also use `#!/usr/bin/python3` on the first line for Python scripts.
- Replace `yourusername` and folder paths with your actual username and target directory.
- In this example, we included an if statement. It starts with `if` and ends with `fi` (literally 'if' spelled backwards). The batch script checks if the said file exists, in that case the condition in the statement is true, so the file will be copied. If the file does not exist, this script will print 'ERROR: example.txt does not exist' on your screen.
- If you want to calculate a MD5 checksum, use: `md5sum "$COPY"`
- Save the batch script as a `.sh` file. To run the file on Linux, open Terminal, navigate to your folder and call the script: `chmod +x /path/to/script.sh && /path/to/script.sh`

Lists

Python is not only a great language to create games in, or to automate boring stuff, it is also very helpful if you want to handle large amounts of data. In this module we will focus on using lists in Python for the Hangman game, but you will see later on that this data type will come in handy for other programs as well. For that reason we will focus on the basics first.

Read and work through Chapter 4 (until 'A Short Program: Conway's Game of Life') of *Automate the Boring Stuff*.

The Hangman Game

Now that you understand the basics of lists and dictionaries, it is time to create the Hangman game. Because this is a more advanced game, you first have to start designing the game in a flowchart, before you can start writing the code for the game.

Create a Flowchart

Hangman starts with a secret word that the player has to guess. Think of all the possibilities to choose from, and the options there are depending on if a character is in the secret word or not.

Create a flowchart for the Hangman game. When you're finished, read and work through Chapter 7 of *Invent Your Own Computer Games with Python*.

Writing the Code

Now that you've created a flowchart and learned how to use `ASCII art` to draw the hangman, it is time to write the code for the game. The first step is to incorporate lists, methods and `elif statements` in your code.

Read and work through Chapter 8 of *Invent Your Own Computer Games with Python*.

Bits and Bots in Practice – Python Exercise

The expert session in September is about "Bits and Bots in practice" and will be given by Eva van den Hurk - van 't Klooster. This part of the module should be done *after* that expert session. We will also make sure to provide you all the slides of the session afterwards to aid you in the assignment.

Abstract expert session

The first Bits and Bots courses in 2024 have provided its members with knowledge that helps them take their work to the next level. While creating games using HTML, CSS and JavaScript, members have gained insights into what a website consists of and what different languages can be used to build a website, its look and feel and interactive features. This helps to improve acquisition and preservation questions when it comes to web archiving; with detailed knowledge of what makes a website it becomes easier gather all the information needed to preserve a website as completely as possible. In this session, we will have a look at these questions.

On the other hand, knowledge of Python can help improve preservation processes. Python scripts can automate the counting of Tweets, change filenames in bulk or deduplicate metadata. This session will touch upon how these last two examples became vital in a large migration project between two digital repositories at Eva's archival institution and how knowledge of Python helped create and understand the R-scripts that were used during this project.

Exercise Bits and Bots in practice

Imagine you have received a ton of metadata from the provider of your previous digital repository. How do you quickly check the file extensions and how do you deduplicate your metadata? With a Python script of course! You will receive a test Excel file to use your Python script on.

For this exercise, you will create a Python script that will help you check several things:

- Check file extensions (in this case, everything needs to be a PDF file)
- Change file names in bulk (replace spaces with underscores)
- Deduplicate PDF files
- Deduplicate BAG building ID's
- Deduplicate BAG residential object ID's
- Deduplicate names of owners
- Deduplicate addresses

This exercise uses the Pandas library. Before starting the exercise, please browse the Pandas Tutorial on W3 Schools.

Steps to follow:

1. Open a new file in IDLE or Mu.
2. For this script, you will need to import several modules. Write the following:

```
import pandas as pd
import os
```

   Remember when writing your script: you've just imported a module. This means that whenever you want to use it, you have to call it. This is similar to what you have done with the Random module (namely the `random.randint()` function).
3. Next, use the correct code to create a path to the location where the file you want to analyze is located, then locate the file.
4. Import the Excel file.
5. Create the code that checks the extension of the files in the Excel. Use the `.str[]`, `.tolist()` and `.drop()` functions and an `if` statement.
6. Then, replace the spaces in the filenames with underscores, using the `.replace function`.
7. Combine columns for the five items you want to deduplicate. Remember: the first column in your Excel is 0, not 1!

```
# 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
```

8. Check for every of the five items whether the metadata can be found multiple times in the Excel file and deduplicate the metadata.

```
# 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
```

9. Save the output file as CSV.

Questions to answer:

- How many files with a different extension than PDF are there in the Excel?
- Where can you see this and where in the file are they?