PHOT 110: Introduction to programming

Project topics: project 2

Michaël Barbier, Spring semester (2023-2024)

Introduction

There are two projects to be completed for the PHOT 110 course during this semester. This file contains the project topics for the **second** project. The projects are strongly focused on one application and will force you to think not only about the Python code itself, but also about the problem-solving aspect.

You can and are encouraged to work together on projects, further, you can ask help from me, Hazan, and Metin (asking help will not influence your project grade). However, your project report and corresponding script should be made individually and not copied from others or online resources. Please also cite any references that you used in the report.

Type of report for project 2

You need to send in both the script(s) used and a report. Please see the example project report: `phot110_example_project_report.docx`. In the report you describe in a concise manner what the result was of your project and how you obtained it. Describe the crucial steps that you undertook to tackle the project task. Support your ideas with plots or schematic drawings. The report should be minimum 1 page, and maximum two pages including figures. The report should be in pdf-format, font-size 12.

For project 2, more emphasis will be given to whether your report is written in an appropriate and formal way. Please check the example report. Further, the coding task should require less time than for the first project, please tell me, Hazan, or Metin whenever you are stuck at a certain step.

Grading of the project

This project will count for 20% of your grade. Points are given on the combined effort of the project and the oral explanation of it during the exam. This means that you will know your final points only after the exam.

Please understand the code/report that you send in, you will be asked questions about it on the final exam. It is better to have a less "fancy" script than not being able to explain the code/report on the exam. The points will be calculated according to the geometric mean of (1) your grade of the code/report and (2) your explanation at the exam. Please ask help to your instructors on time, we might have not enough time to help you at the last day before the deadline of the report.

The total of projects made during the semester counts for 40% of your total grade for the course.

Project topics

Next is a list of 3 topics you can choose out for your project together with their task description.

Find your way using OpenStreetMap

A GPS (or web applications such as Google Maps) can show us the shortest road from a start point to a destination (or the road which would take us the least time). The algorithms to find the road from start to destination therefore looks at all possible paths towards the destination and chooses the shortest one. Here we look into how we can access OpenStreetMap data in Python, and find a route between two points.

First look at OpenStreetMap data via the Overpass Turbo website and fill in the following region: "Gülbahçe, Izmir, Aegean region, Turkey" to zoom to a small region of interest. If you see only a small region (for example Gülbahçe) you can ask for all the "highway" data elements:

```
[out:json][timeout:25];
way["highway"]({{bbox}});
out geom;
```

Remark that these "highway" data elements represent all type of roads, also pedestrian roads, etc. Export the data to for example Geojson format. Open the file in a text editor.

Next, automate this process in Python. You can do this using the Python package OSMnx.

- Install the Python OSMnx package via pip or PyCharm
- Look at a simple guide to use the OSMnx package.
- Then use it to download OpenStreetMap road data from Gülbahçe as a graph.
- Afterwards download the data of the buildings in the same region.

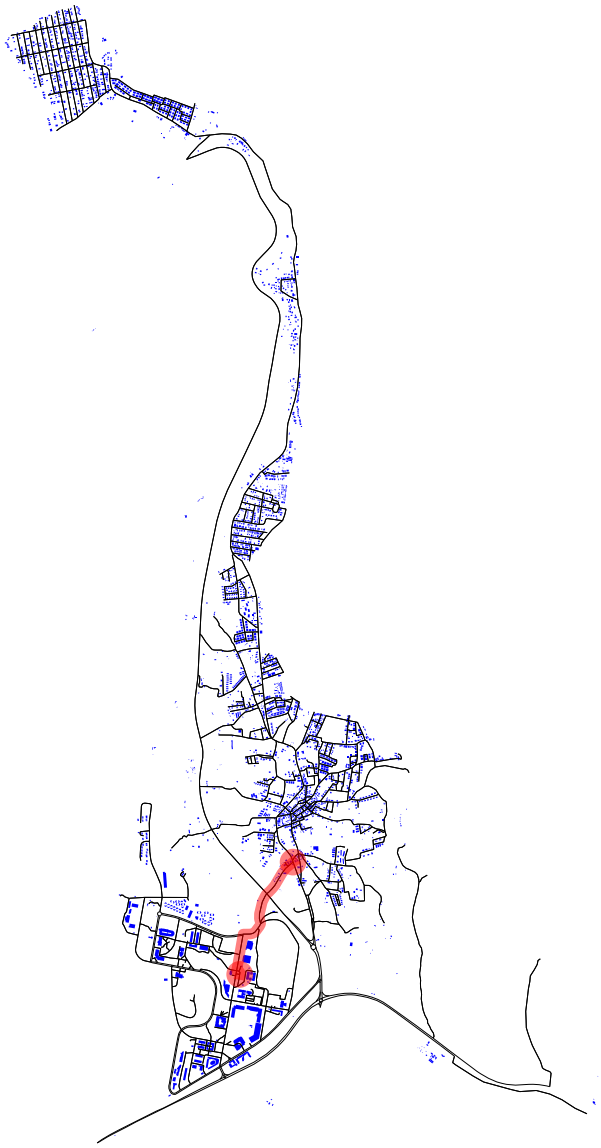
```
import matplotlib.pyplot as plt
# Import the osmnx library
import osmnx as ox
# Networkx library to calculate shortest paths
import networkx as nx

# Download the road data of a small region from OpenStreetMap,
# here we download the data for Gülbahçe.
place_name = "Gülbahçe, Izmir, Aegean region, Turkey"
g = ox.graph_from_place(place_name)
buildings = ox.features_from_place(
  place_name, tags={'building': True})
```

- You can find more examples on the GitHub repository as Jupyter Notebooks.
- Calculate the route between two favorite points on the map, you can use OpenStreetMap to get the longitude and latitude coordinates. This example code shows the shortest route from the grass in front of the Photonics building to the BIM supermarket.

```
# Route between points of interest
# Points in (longitude, latitude)
origin_point = (26.6398892, 38.3209038)
destination_point = (26.6440502, 38.3285891)
# Find the nearest "road" node of the graph
origin_node = ox.nearest_nodes(
    g, origin_point[0], origin_point[1])
destination_node = ox.nearest_nodes(
    g, destination_point[0], destination_point[1])
# Calculate the shortest path between the nodes
route = nx.shortest_path(
    g, origin_node, destination_node, weight='length')
```

You can plot the various features: roads, buildings, and the route, together like in the following example code:

```
# Plot the buildings, roads, and the calculated route
fig, ax = ox.plot_footprints(
    buildings, alpha=1, bgcolor="white", color="blue", show=False)
fig, ax = ox.plot_graph(
    g, ax=ax, node_size=0, edge_color="black", edge_linewidth=0.2,
    show=False)
fig, ax = ox.plot_graph_route(
    g, route, ax=ax, route_color="red", route_linewidth=4, node_size=2)
plt.show()
```



- [optional] Try to extract your favorite road (e.g. “Gülbahçe caddesi”) from the road geometries, together with your favorite building. The `buildings` object obtained from the example code above is already in geopandas (table) format. For the roads you can convert the graph object `g` to a geopandas table for that:

```
gdfs = ox.graph_to_gdfs(g)
```

Encryption tool

Encryption of messages and files is used to securely transporting data. Simple encryption algorithm use a single “symmetric” key which is both used for encryption and decryption. This type of method is insecure as it requires the sender to share the key with the receiver somehow (in a secure manner) up front. To avoid this problem, an asymmetric approach

can be used, that is, using a different keys for encryption and decryption. RSA is such an encryption algorithm and is much more secure. Nevertheless, remember that secure algorithms are still vulnerable due to human factors as illustrated by the artists of xkcd comics.

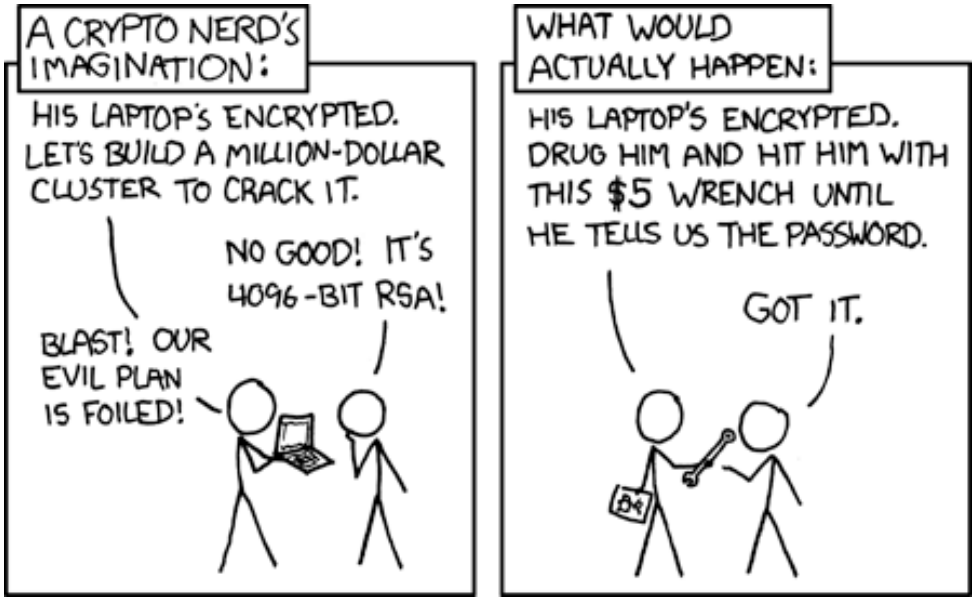


Figure 1: Cartoon taken from xkcd (https://xkcd.com/538/)

The idea of the RSA encryption algorithm is that there are two keys instead of one: one is used by the sender (the public key), and one is used by the receiver (the private key). Both keys have the relation that a message **encrypted** with **one of them** can be **decrypted** with **the other one** but NOT with the same key. The following procedure is used to send a secure message:

1. **Key generation step**: The receiver generates both the public and the private key and gives the public one to the sender (via a potentially insecure channel).
2. **Encryption**: The sender encrypts the message (or file) with this key.
3. **Transmission** of the message: The sender sends the encrypted message to the receiver via a potentially insecure channel.
4. **Decryption**: The receiver uses the private key to decrypt the message.

In this project you will implement the RSA algorithm in Python to improve the security. For this you can perform the following steps:

- First follow the explanation of the RSA encryption scheme to understand how the RSA algorithm can be implemented in Python.
- Next, try out the Python "rsa" package, see for example the cryptography howto in GeekforGeeks to understand how to use it.
- Test your encryption by encrypting first a string.
- Extend your script to enable the encryption and decryption of files.
- Show that the encryption works by using it on this pdf-file.
- How long does the generation of a 512-bit RSA key take?
- Does it take longer to encrypt a file with a larger key?

Lossless file compression

File compression is very often employed to reduce the disk space that a file occupies on your computers hard disk. Or maybe even more important when sending a file over the network or internet. Various compression methods exist, both lossy (losing file quality) and lossless methods.

Within this project we will employ at the zlib Python package, to understand some of the aspects of lossless file compression. This package provides compression using the DEFLATE algorithm which combines LZ77 compression with Huffmann coding:

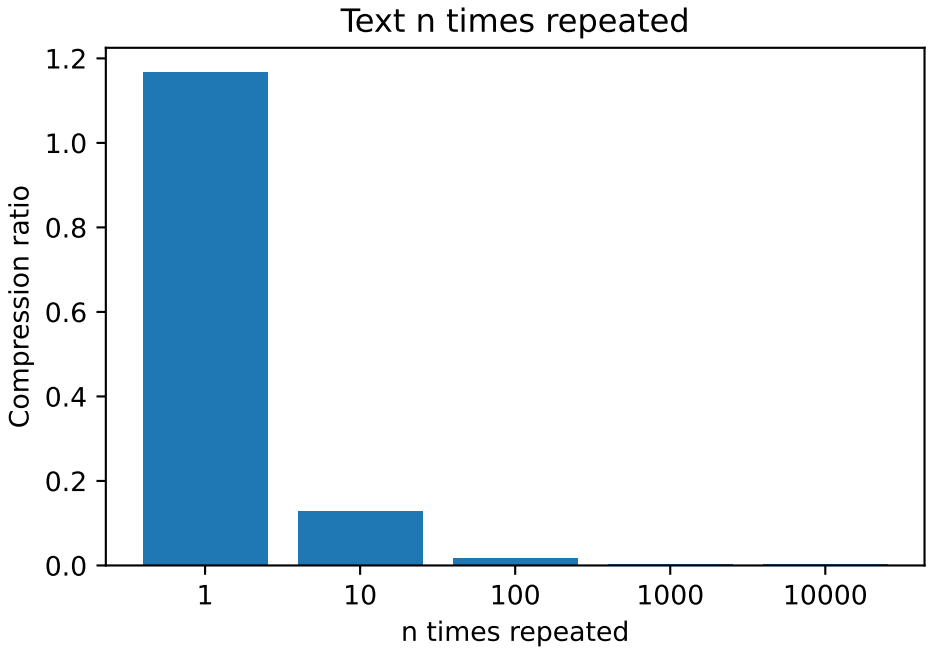
- LZ77 compression: rename sequences of repeated bytes, for example, if you have the string `Hellooooooooooooooooooo!` then the last sequence of `o`'s could be encoded as <20 times "o">
- Huffmann encoding: replaces frequently encountered bytes/characters by shorter codes, for example, in the string `ABBBBABABBOBABBABOBBB` the `B` characters are more frequent and could be encoded by a single bit reducing the space they need.

See this explanation on different DEFLATE algorithm implementations.

Within this project the goal is to perform the following steps:

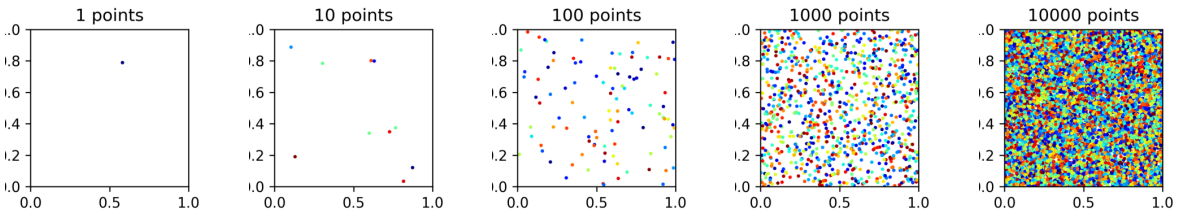
- Make a script that allows you to compress a string using the zlib Python package. You can look at this usage example of zlib in Python.
- Compress a sentence using your implementation
- Compute the compression ratio, in the case of a string: the ratio of the number of bytes (or characters) of the compressed string vs the original string.
- Decompress it, do you get the original sentence back?
- Then compress a plot you made with Matplotlib and saved as ".tif" file.
- Compute the compression ratio: that is, the ratio of the compressed file size over the original file size.
- Compress the image with your script, how efficient is the compression, is it better than compressing text? Why is that?
- Why is a text where you repeat a sentence over and over, more compressible than random text? Test this.

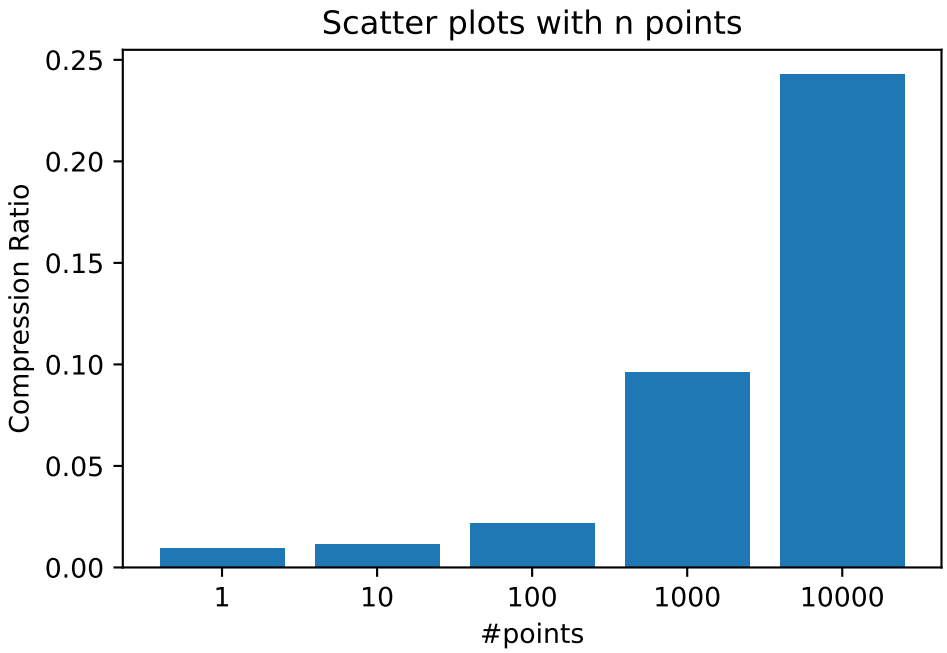
```
The sentence is:
Repeating text contains less information.

Sentence repeated n = 1 times: compression ratio = 1.1667
Sentence repeated n = 10 times: compression ratio = 0.1286
Sentence repeated n = 100 times: compression ratio = 0.0183
Sentence repeated n = 1000 times: compression ratio = 0.0045
Sentence repeated n = 10000 times: compression ratio = 0.0031
```



- Do a similar test but for different types of images, for example as below where we show scatter plots with different numbers of points. Basically you want to test the compression on images with different amount of "information".

```
Image with n = 1: compression ratio = 0.0096
Image with n = 10: compression ratio = 0.0113
Image with n = 100: compression ratio = 0.022
Image with n = 1000: compression ratio = 0.0964
Image with n = 10000: compression ratio = 0.2428
```




Scatter plots with n points
Compression Ratio
0.25
0.20
0.15
0.10
0.05
0.00
1
10
100
1000
10000
#points