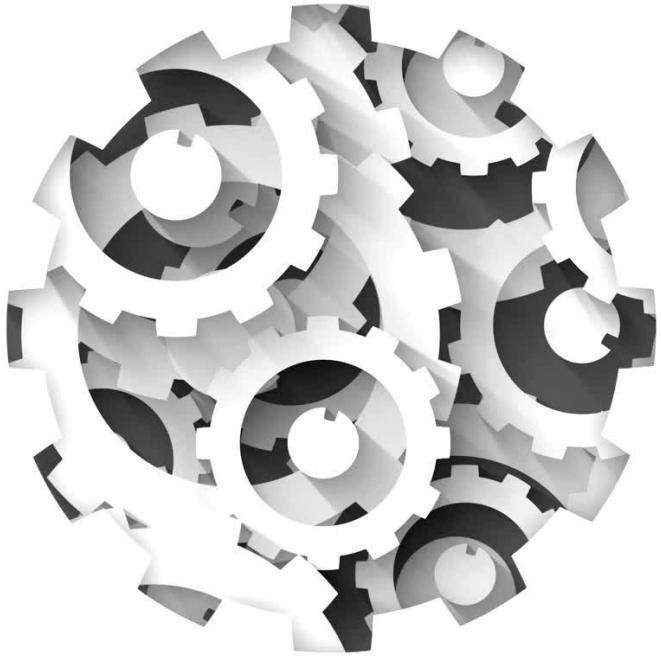
SUBJECT FOCUS DAY

Leeds Trinity University



Python libraries

Content

- Use of Python libraries

Quick Response (QR) Code

A **QR-code** is a two-dimensional bar code used for its fast readability and comparatively large storage capacity.

Python has a library "**QRCODE**" for generating QR code images. It can be installed using pip.

Approach:

- Import module
- Create Qrcode that embed your information with **qrcode.make()**
- Save into image

Python Platform

For today's activity we will use REPLIT to run ALL our computer codes.

- REPLIT https://replit.com/

First Activity:

QRCODE

Get a QR_Code

```
import qrcode

# Data to be encoded
data = 'My name is __, I like Leeds Trinity University'

# Encoding data using make() function
qr_img = qrcode.make(data)

# Saving as an image file
qr_img.save('MyQRCode1.png')
```

Example 2: Get a QR_Code (Website)

```
import qrcode

# Data to be encoded
data = 'https://www.leedstrinity.ac.uk/'

# Encoding data using make() function
qr_img = qrcode.make(data)

# Saving as an image file
qr_img.save('MyQRCode2.png')
```

Example 3: Customised QR Code

```
import qrcode

data = “My destination this September is LTU "
# Creating an instance of QRCode class
qr = qrcode.QRCode(version = 1,
                   box_size = 10,
                   border = 5)

# Adding data to the instance 'qr'
qr.add_data(data)

qr.make(fit = True)
img = qr.make_image(fill_color = 'red', back_color = 'white')

img.save('MyQRCode2.png')
```

Activity

I. Create any message of your choice and encode it in a QRCode.

II. Change the colour of the Qrcode

Python Fake

- Using the fake library to create a dummy data.

Creating Fake Data

- Python has a library that can help you to create fake or dummy data for research and demonstration purpose.

- Consider this as a sample data, to avoid violating user's privacy.

<u>Approach</u>:

```
Import module  (pip install Faker) or add as package.
Use the Faker method to create the object.
Then call modules that have names, addresses etc
```

Creating Fake Data

```
from faker import Faker
#fake names
fake = Faker()
print("My sample data: ", fake.name())    #'Paul Lynn'
print(fake.name()) #'Keith Soto'

#fake address
print("My sample address: ", fake.address())
'Unit 6944 Box 5854\nDPO AA 14829'

print("My sample address: ", fake.address()
'44817 Wallace Way Apt. 376\nSouth Ashleymouth, GA 03737'
```

Creating Fake Data

```
Create a fake dictionary
print("My sample dictionary: ", fake.pydict())

Create a fake list
print("My sample list: ",fake.pylist()

Create a fake int
print("My sample integer: ",fake.pyint()

Create a fake float
print("My sample float: ", fake.pyfloat())

Create a fake str
print("My sample text: ",fake.text())
```

Create 10 Sample Data of names

```
for i in range(10):
  print(fake.name())

#Let us localise the name, for example italian names
fakeItalian = Faker('it_IT')
for k in range(10):
    print(fakeItalian.name())

#Faker('ja_JP') #(bg_BG,cs_CZ,zh_CN,zh_TW)
#Faker('en_US') #("de_DE")
```

[Locale ar_AA — Faker 13.3.2 documentation]

[List of country codes by alpha-2, alpha-3 code (ISO 3166) (iban.com)]

Create Mixture of Sample Data

```
fake = Faker(['bg_BG', 'de_DE', 'cs_CZ'])

for _ in range(10):
    print(fake.name())
```

Use –l to see the languages.

Activity

- Select a country and create a dummy data of names and addresses for that country.

PyShortener

Shorten a Link

URL shortener reduces a long link.It is useful for sharing long links in documents and presentations in a short and compact link.

pyshorteners is a Python lib to help you short and expand urls using the most famous URL Shorteners availables.

Approach:

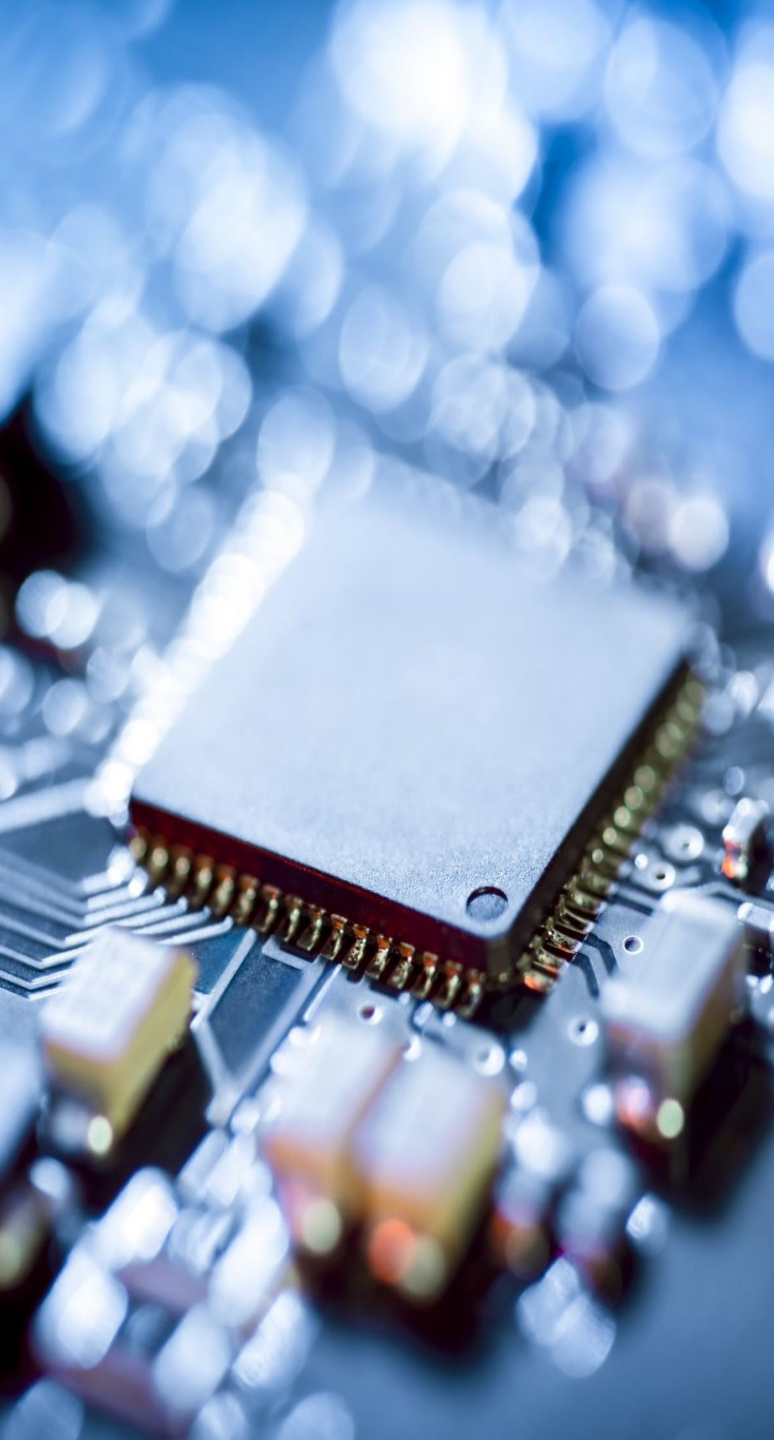
1. Install the package
2. Import the package
3. And call the pyshortener module

Pyshortener

```
pip install pyshorteners   //or add as package in REPL

import pyshorteners

#invoke the pyshortener object
myurl= pyshorteners.Shortener()

#using the object, call the tinyurl method and pass the link
print(myurl.tinyurl.short('https://www.marvel.com/'))
```

Pyshortener with input

```
import pyshorteners

#call a link
link= input("Please enter a link: ")

create = pyshorteners.Shortener()

#using the object, call the tinyurl method and pass the link
x =create.tinyurl.short(link)

print(x)
```



Calculating your CPU RAM and Cores

Know your computer's RAM amount and CPU counts with these two libraries.

1. Multiprocessing
2. PSutil

<u>Steps</u>:

Add the package

Call the relevant object

Multiprocessing

```
import multiprocessing
print(multiprocessing.cpu_count())
```

PSUtil Library (CPU)

Info about CPU

```
import psutil
print("My cpu times",psutil.cpu_times())
print("My cpu counts", psutil.cpu_count())
print("Idle cpu counts",psutil.cpu_count(logical=False)
#Number of idle CPU
print("My cpu stats",psutil.cpu_stats())          #CPU usage
print("My cpu freq",psutil.cpu_freq())
print("My cpu load",psutil.getloadavg()) #avg system load
over time
```

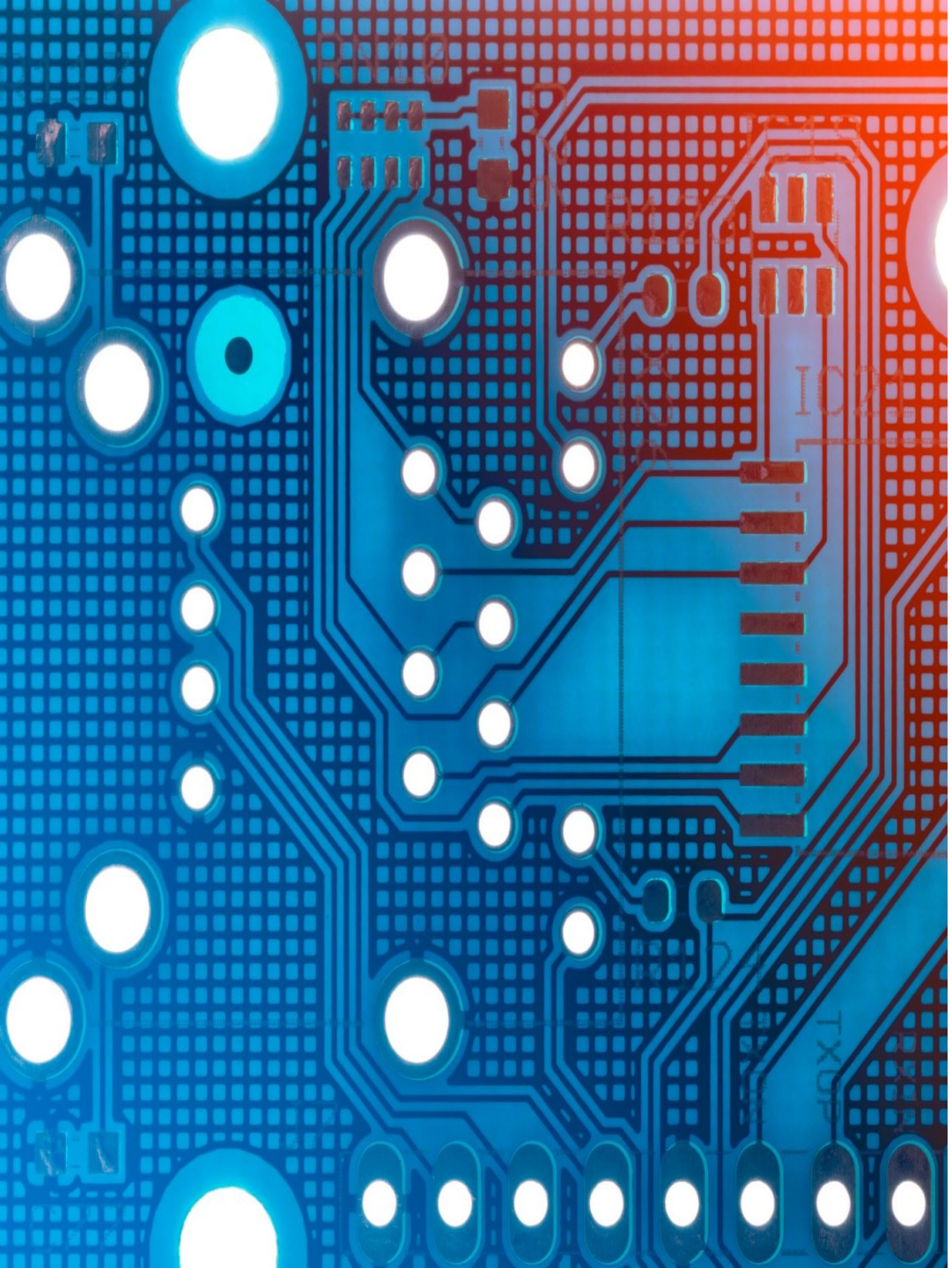
PSUtil Library (Memory)

Info about Memory

```
import psutil
print('RAM memory % used:', psutil.virtual_memory()[2])

print("Available virtual memory", psutil.virtual_memory().available)

print("My total virtual memory ", psutil.virtual_memory().total)

print("My virtual memory in kilobyte", psutil.virtual_memory().total /(1024 ** 1))

print("My virtual memory in megabyte",psutil.virtual_memory().total /1024/1024)

print("My virtual memory in gigabyte ",psutil.virtual_memory().total /(1024 ** 3))

psutil.swap_memory()          //A space used by OS to store RAM info
```

Calculating the space in your Hard Drive

```
hdd = psutil.disk_usage('/')
print("Total space", hdd.total / (1024.0 ** 3))
print("Used space", hdd.used / (1024.0 ** 3))
print("Free space", hdd.free / (1024.0 ** 3))
print("Percentage of used", hdd.percent / (1024.0 **
3))

Print(hdd)
```



Activity

- Print some useful CPU information

Take Home Activity

- Publish your App and share it with your friends.

From .py to .exe

We want to share our python project with friends, family and the public.

Steps in making a .py to exe

1. **Install the python library *pyinstaller***
2. Navigate to the directory of your '.py' file.
3. Copy the path
4. Open powershell
5. Use python installer and run the file
6. Go to distribution folder and get your application (.exe)

From .py to exe

Step 1: Install the library *pyinstaller* in powershell.

```
pip install pyinstaller
```

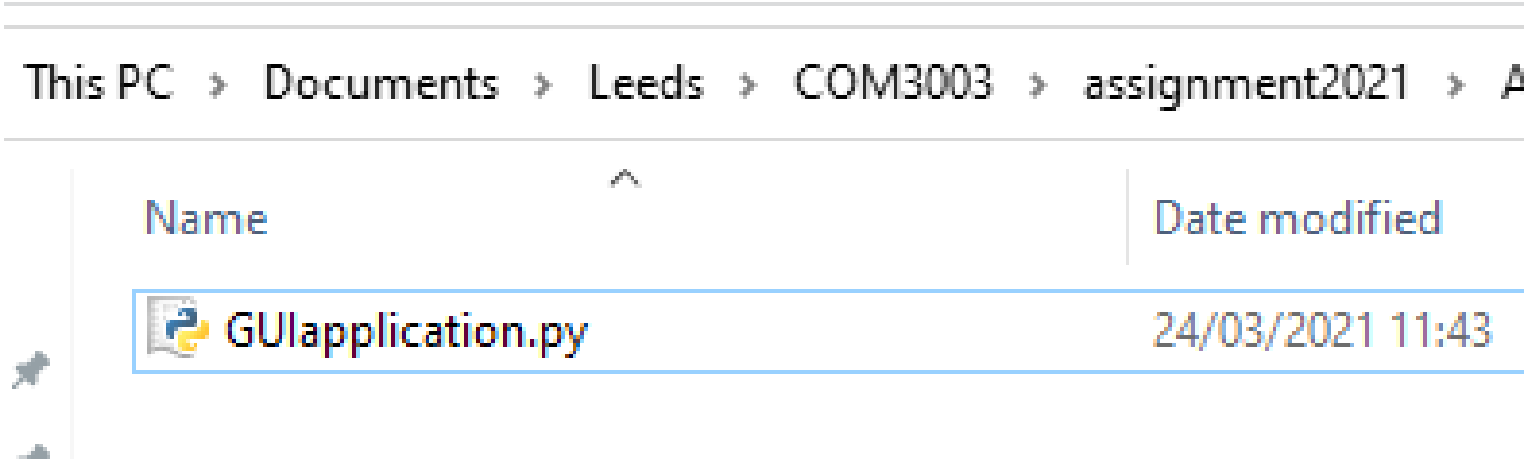
Go to your programs section in C: drive and type this command

C:\users\id916438\appdata\local\programs\python\python37>

```
python.exe  -m pip install --upgrade pip
```

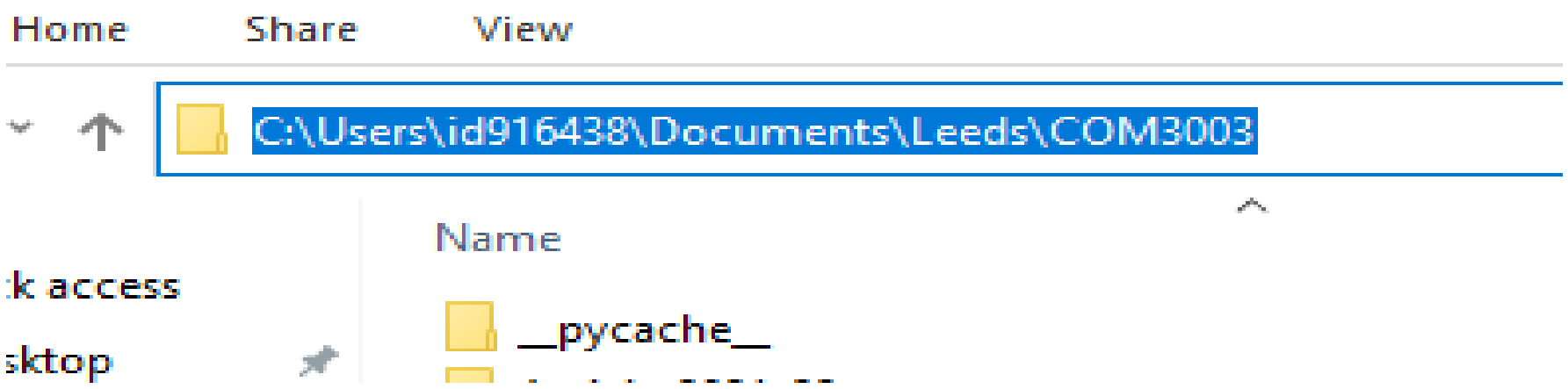
Making python exe file

Step 2:
Navigate to the directory of your '.py' file.



From .py to exe (Go to powershell)

3. Go to the window location tab and copy the path or press shift and right click, then choose "open powershell window here".



4. Open powershell and type "cd [your tab location]". Example

```
cd C:\Users\id916438\Documents\Leeds\COM3003
```

From .pye to exe

5. Type this command in powershell.

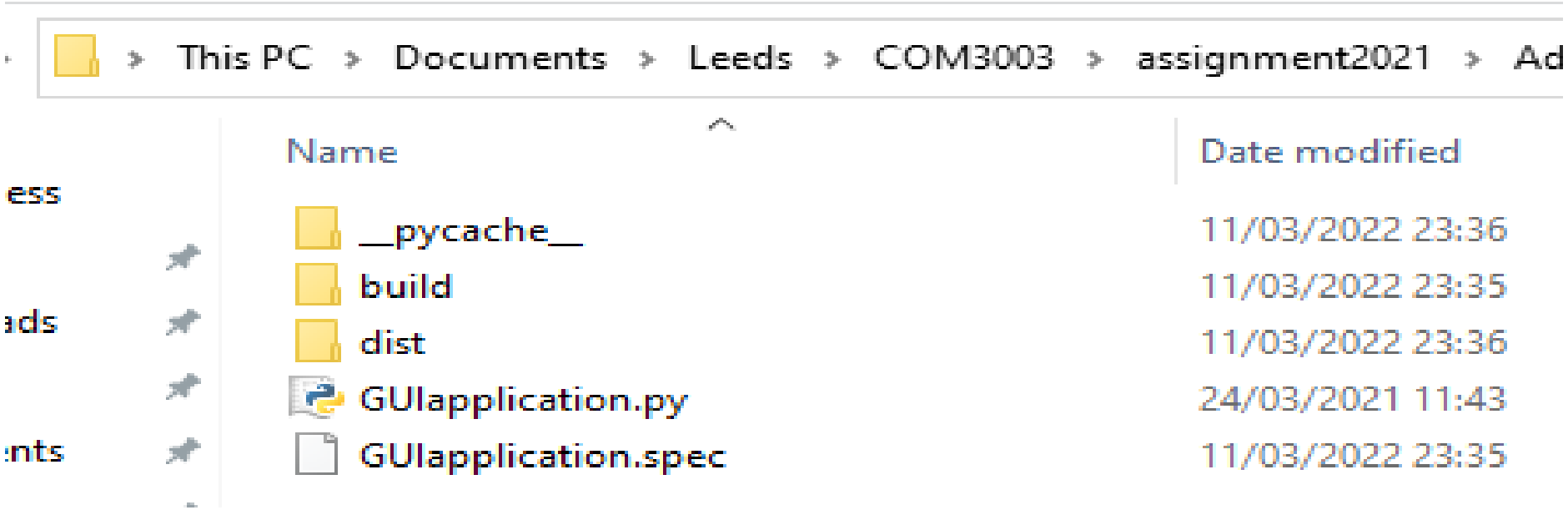
- pyinstaller --onefile 'filename.py'

```
34260 INFO: Building PYZ (ZlibArchive) C:\Users\id916438\Documents\Leeds\COM3003\assignment
plication\PYZ-00.pyz
35015 INFO: Building PYZ (ZlibArchive) C:\Users\id916438\Documents\Leeds\COM3003\assignment
plication\PYZ-00.pyz completed successfully.
35041 INFO: checking PKG
35041 INFO: Building PKG because PKG-00.toc is non existent
35042 INFO: Building PKG (CArchive) GUIapplication.pkg
71580 INFO: Building PKG (CArchive) GUIapplication.pkg completed successfully.
71636 INFO: Bootloader c:\users\id916438\appdata\local\programs\python\python37\lib\site-pa
\runw.exe
71636 INFO: checking EXE
71638 INFO: Building EXE because EXE-00.toc is non existent
71640 INFO: Building EXE from EXE-00.toc
71641 INFO: Copying bootloader EXE to C:\Users\id916438\Documents\Leeds\COM3003\assignment2
ication.exe.notanexecutable
73156 INFO: Copying icon to EXE
73157 INFO: Copying icons from ['c:\\users\\id916438\\appdata\\local\\programs\\python\\pyt
loader\\images\\icon-windowed.ico']
73437 INFO: Writing RT_GROUP_ICON 0 resource with 104 bytes
73438 INFO: Writing RT_ICON 1 resource with 3752 bytes
73441 INFO: Writing RT_ICON 2 resource with 2216 bytes
73445 INFO: Writing RT_ICON 3 resource with 1384 bytes
73446 INFO: Writing RT_ICON 4 resource with 38188 bytes
73446 INFO: Writing RT_ICON 5 resource with 9640 bytes
73447 INFO: Writing RT_ICON 6 resource with 4264 bytes
73448 INFO: Writing RT_ICON 7 resource with 1128 bytes
73455 INFO: Copying 0 resources to EXE
73455 INFO: Emedding manifest in EXE
73457 INFO: Updating manifest in C:\Users\id916438\Documents\Leeds\COM3003\assignment2021\A
on.exe.notanexecutable
73612 INFO: Updating resource type 24 name 1 language 0
73624 INFO: Appending PKG archive to EXE
78802 INFO: Building EXE from EXE-00.toc completed successfully.
PS C:\Users\id916438\Documents\Leeds\COM3003\assignment2021\Adnrea\GUI application\GUI>
```

From .pye to exe

6. After the installation

- You will see a folder called *dist*, open the folder and the app will be there and ready.



From .py to exe

- This is the app in exe form.

Name	Date modified	Type	Size
GUIapplication.exe	11/03/2022 23:36	Application	8,509 KB

Add a library to your exe package

```
pyinstaller --hidden-import 'package_name' --onefile 'filename.py'
```

```
pyinstaller --hidden-import 'phonenumbers' --onefile 'GUIapplication.py'
```

- Then run the packaging command again:

```
pyinstaller --onefile  'filename.py'
pyinstaller --onefile  'hello.py'
```