INTRODUCTION TO PYTHON

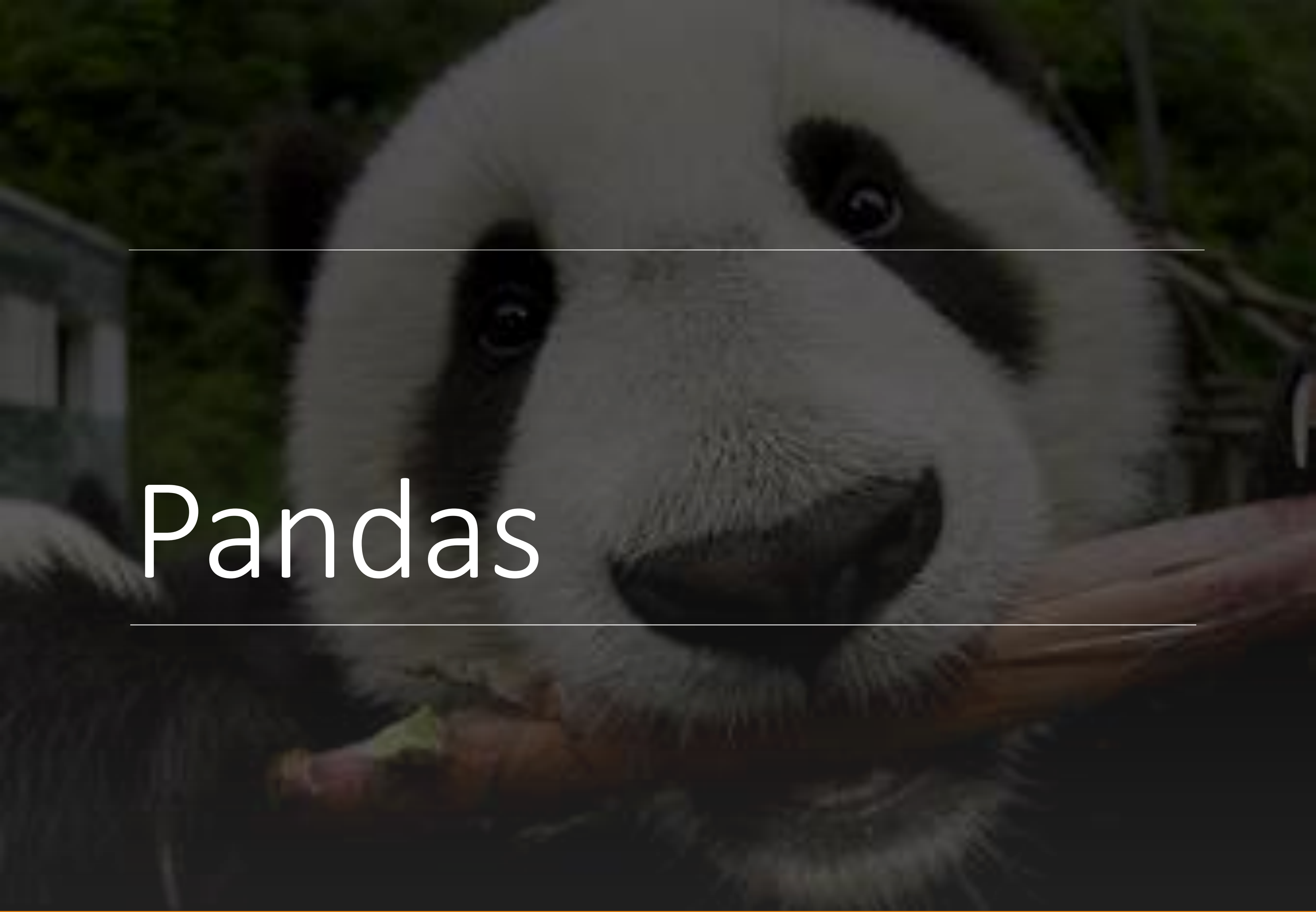
LECTURE 7

Asem Elshimi

Outline

Pandas

Optimization problem



Pandas

Pandas

(**P**ython **an**d **D**ata **A**nalysi**s**)

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html#min

A fast and efficient **DataFrame** object.

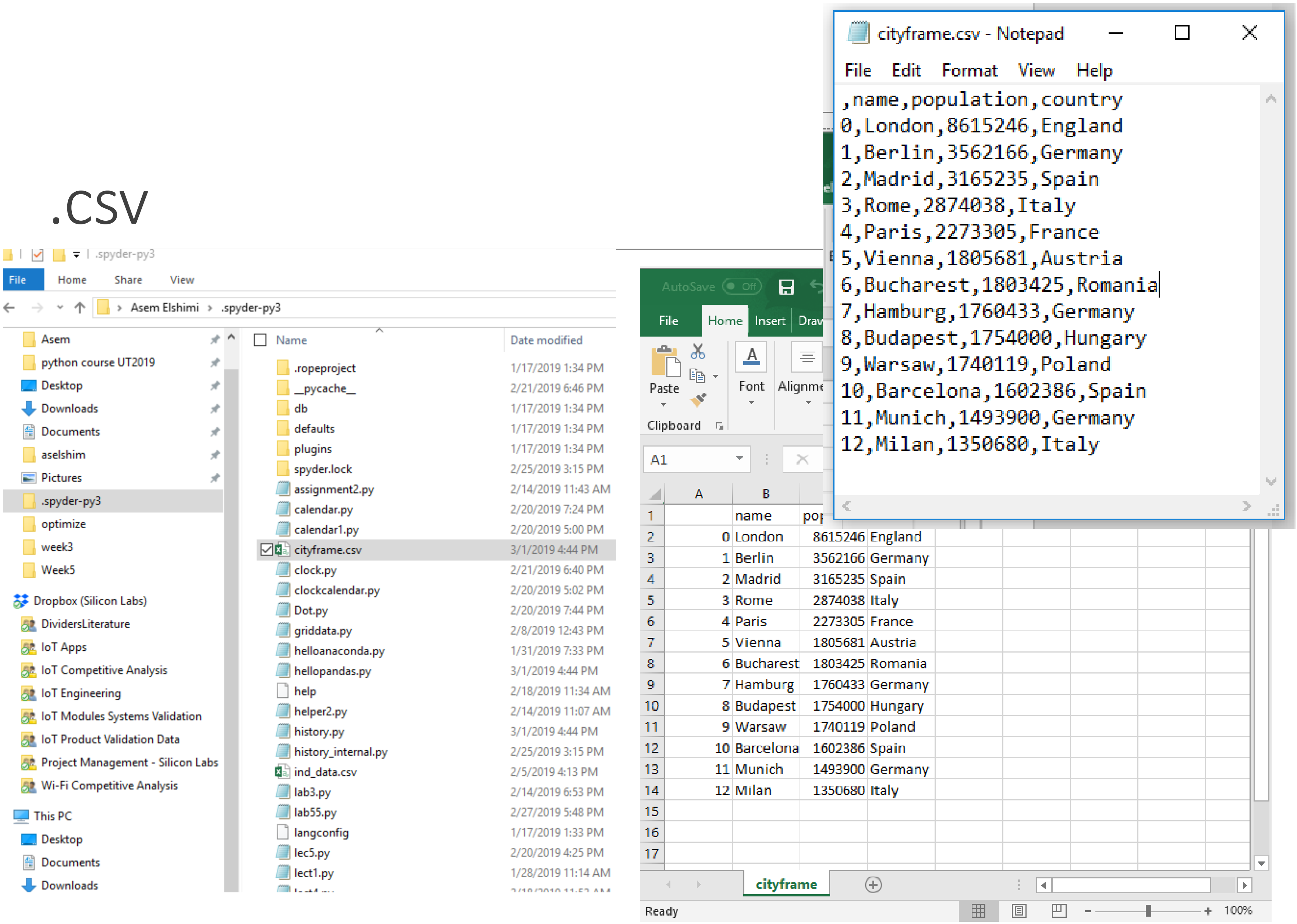
reading and writing data

Flexible **reshaping, slicing**, **fancy indexing**

Python with *pandas* is in use in a wide variety of **academic and commercial** domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

```
import pandas as pd
```

.CSV



```
,name,population,country
0,London,8615246,England
1,Berlin,3562166,Germany
2,Madrid,3165235,Spain
3,Rome,2874038,Italy
4,Paris,2273305,France
5,Vienna,1805681,Austria
6,Bucharest,1803425,Romania
7,Hamburg,1760433,Germany
8,Budapest,1754000,Hungary
9,Warsaw,1740119,Poland
10,Barcelona,1602386,Spain
11,Munich,1493900,Germany
12,Milan,1350680,Italy
```

	A	B		
1		name	pop	
2	0	London	8615246	England
3	1	Berlin	3562166	Germany
4	2	Madrid	3165235	Spain
5	3	Rome	2874038	Italy
6	4	Paris	2273305	France
7	5	Vienna	1805681	Austria
8	6	Bucharest	1803425	Romania
9	7	Hamburg	1760433	Germany
10	8	Budapest	1754000	Hungary
11	9	Warsaw	1740119	Poland
12	10	Barcelona	1602386	Spain
13	11	Munich	1493900	Germany
14	12	Milan	1350680	Italy
15				
16				
17				

Name	Date modified
.ropeproject	1/17/2019 1:34 PM
__pycache__	2/21/2019 6:46 PM
db	1/17/2019 1:34 PM
defaults	1/17/2019 1:34 PM
plugins	1/17/2019 1:34 PM
spyder.lock	2/25/2019 3:15 PM
assignment2.py	2/14/2019 11:43 AM
calendar.py	2/20/2019 7:24 PM
calendar1.py	2/20/2019 5:00 PM
☑ cityframe.csv	3/1/2019 4:44 PM
clock.py	2/21/2019 6:40 PM
clockcalendar.py	2/20/2019 5:02 PM
Dot.py	2/20/2019 7:44 PM
griddata.py	2/8/2019 12:43 PM
helloanaconda.py	1/31/2019 7:33 PM
hellopandas.py	3/1/2019 4:44 PM
help	2/18/2019 11:34 AM
helper2.py	2/14/2019 11:07 AM
history.py	3/1/2019 4:44 PM
history_internal.py	2/25/2019 3:15 PM
ind_data.csv	2/5/2019 4:13 PM
lab3.py	2/14/2019 6:53 PM
lab55.py	2/27/2019 5:48 PM
langconfig	1/17/2019 1:33 PM
lec5.py	2/20/2019 4:25 PM
lect1.py	1/28/2019 11:14 AM

Pandas vs excel!

Analyze large datasets:

- Excel is sluggish at 10000 rows

More high level functions.

More file formats: CSV, HTML, SQL.

Automated procedures.

Co-existence!

Extra-reads

Pandas visualization:

- https://towardsdatascience.com/the-art-of-effective-visualization-of-multi-dimensional-data-6c7202990c57

Web scraping:

- https://realpython.com/python-web-scraping-practical-introduction/

Matplotlib and code blocking:

- https://stackoverflow.com/questions/28269157/plotting-in-a-non-blocking-way-with-matplotlib/33050617

Python Scripting

Python scripting

Programs well designed to be launched by other programs become more powerful than their code alone.

You can read more about Unix philosophy at https://en.wikipedia.org/wiki/Unix_philosophy/ .

File handling I/O, CSV

```
import csv
import pandas as pd
import datetime

now = datetime.datetime.now()
result_dir = "./tempas/"
test_name="hello_csv"
fname = result_dir + now.strftime("%Y_%m_%d_%H_%M_") + test_name + ".csv"

import os.path
if not (os.path.isfile(fname)): #if no recordings at all
    with open(fname,mode='w',newline='') as wfile:#create a new file
        header = ["brd", "temp", "pa_mode"]
        csv_writer=csv.writer(wfile)
        csv_writer.writerow(header)

#writing to csv file
with open(fname,mode='a',newline='') as wfile:
    csv_writer=csv.writer(wfile)
    csv_writer.writerow([1,2,3])
    csv_writer.writerow([4,5,6])
    csv_writer.writerow([4,2,6])


#reading the entire csv file
df=pd.read_csv(fname)
print(df)
```

Making files executable

Specify the interpreter:

- Shebang
- #! Python_directory

Make file executable:

- Chmod +x python_file

Command line arguments

```
#!/designtools/python_3.6.5/bin/python3.6

import sys


for x in range(len(sys.argv)):
     print ("Argument: ", sys.argv[x])
```

Launching other programs

```
#!/designtools/python_3.6.5/bin/python3.6
import subprocess
print('About to run ls.')
subprocess.call(['ls', '-l'])
print('Finished running ls.')
```

Return code

```
import subprocess
print('About to run ls.')
rc = subprocess.call(['ls', '-l'])
print('Finished running ls.')
print('RC = {:d}'.format(rc))
```

```
0 1 or 127(not found)
```

check_output

```python
import subprocess

ls_output_raw = subprocess.check_output(['ls', '-l'])

ls_output_text = ls_output_raw.decode('UTF-8')

print(ls_output_text)
```

Running multiple subprocesses?

```
p = sp.Popen(['ls', '-l'])
rc=p.wait()
print(rc)
```

Read, write, and interact:

https://pymotw.com/2/subprocess/

Inductors

Z=jwL

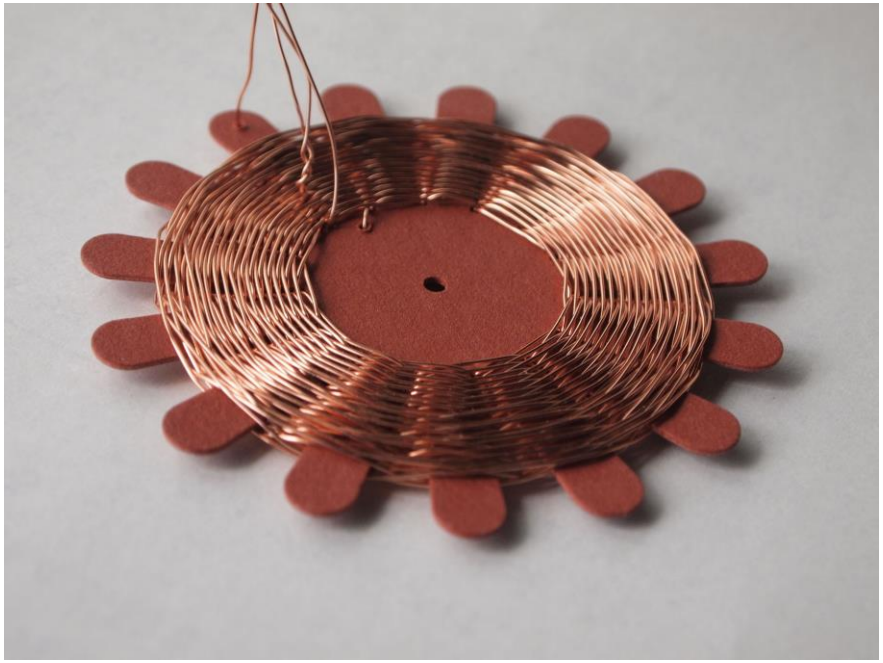
Electric current→Magnetic field→induced currents

Energy transformers

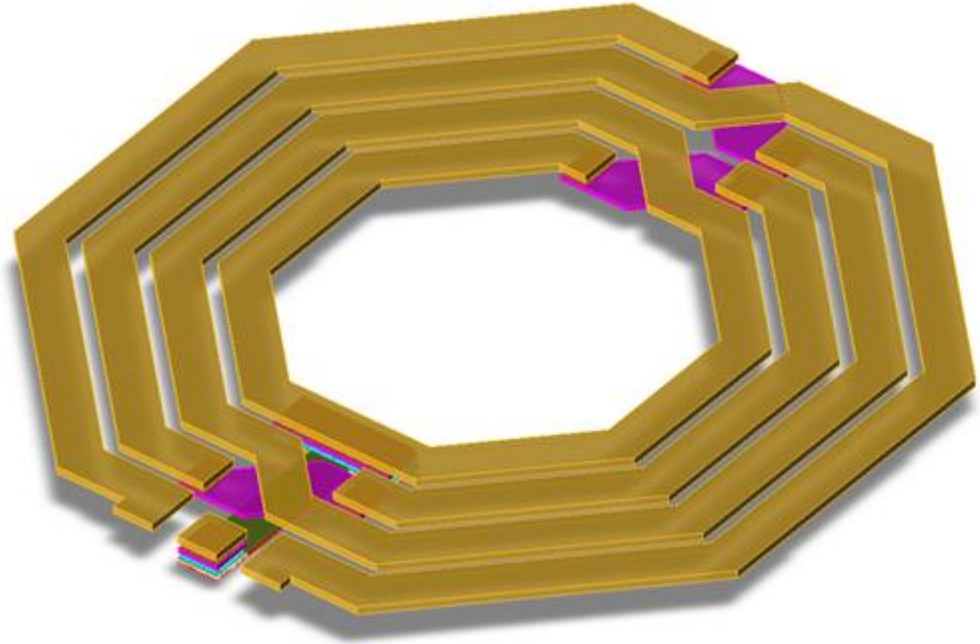
RF filters

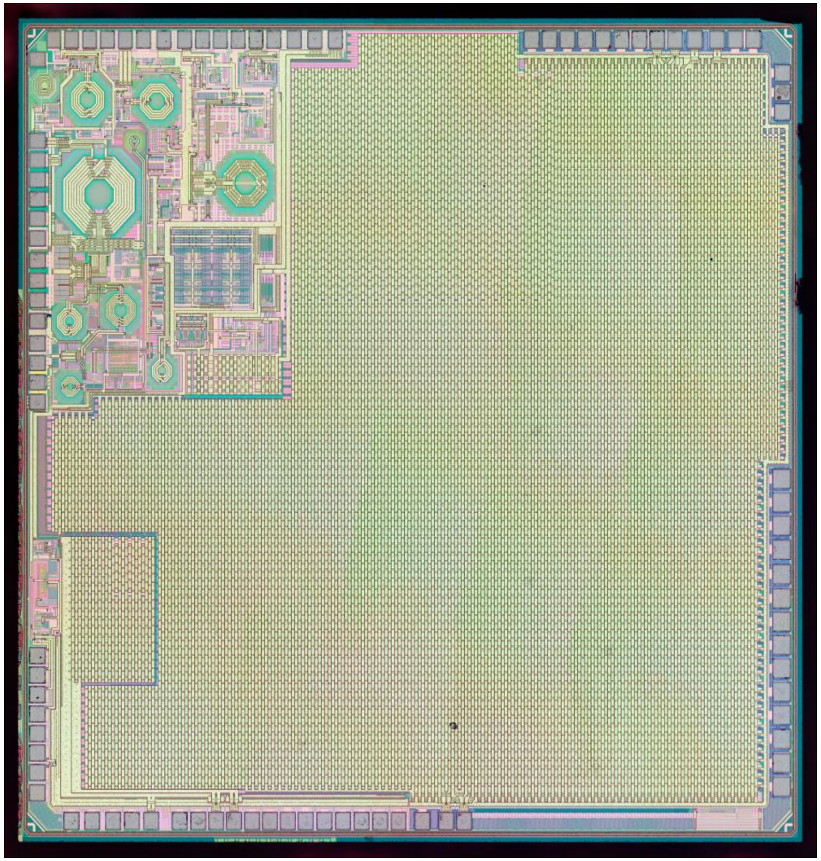
RF resonators

Power lines

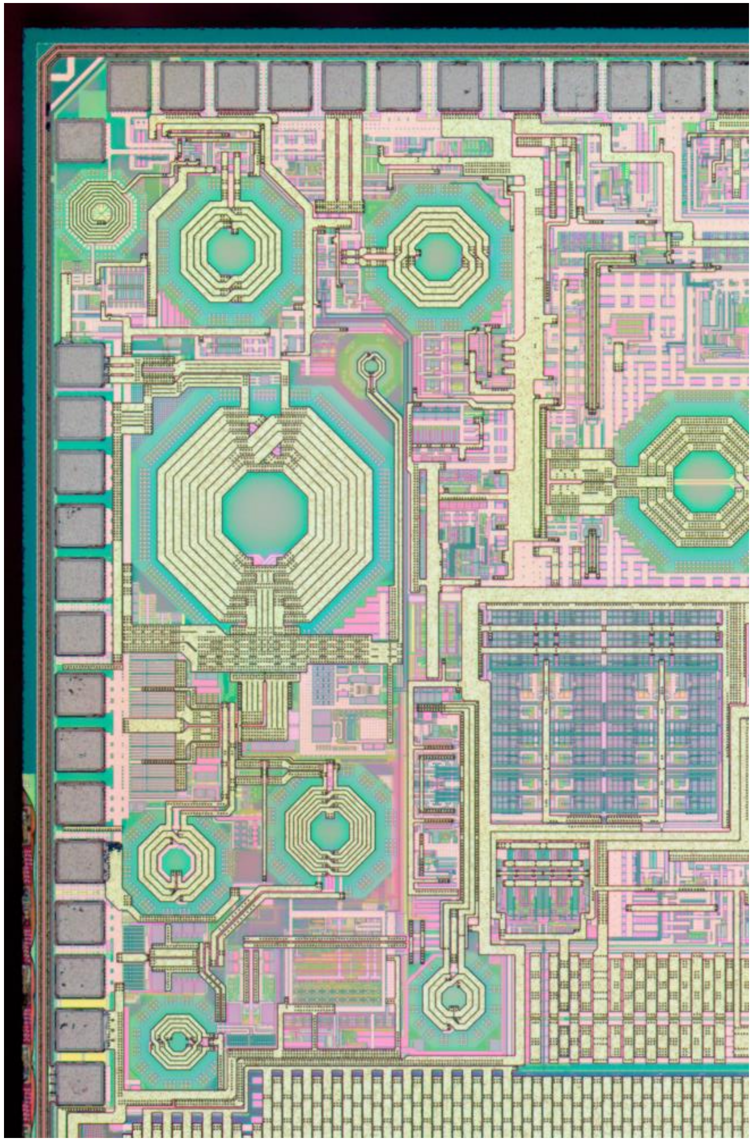


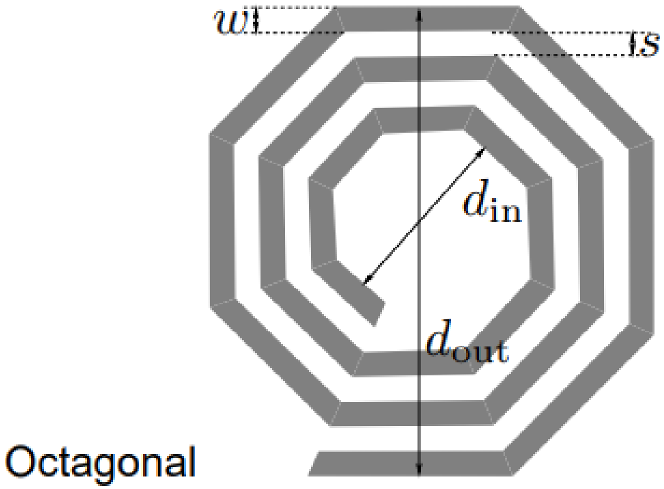
On chip inductors

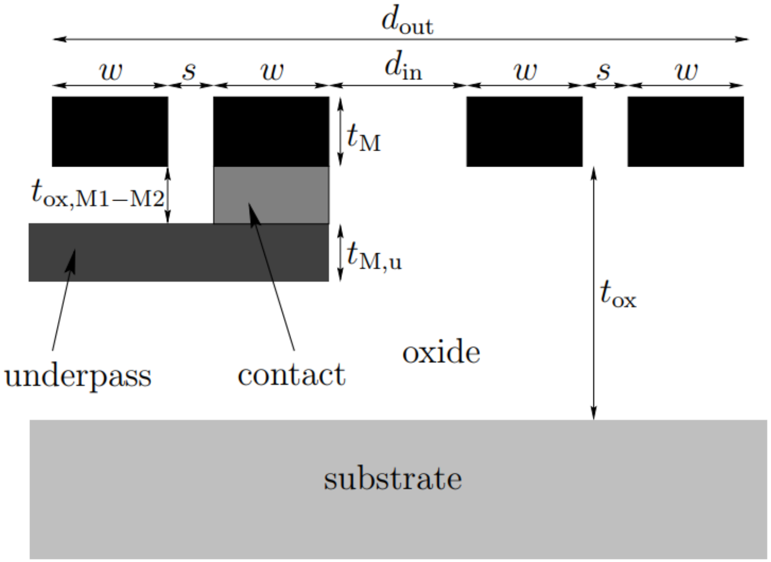




https://zeptobars.com/en/read/Espressif-ESP32-Wi-Fi-Bluetooth-2.4Ghz-ISM

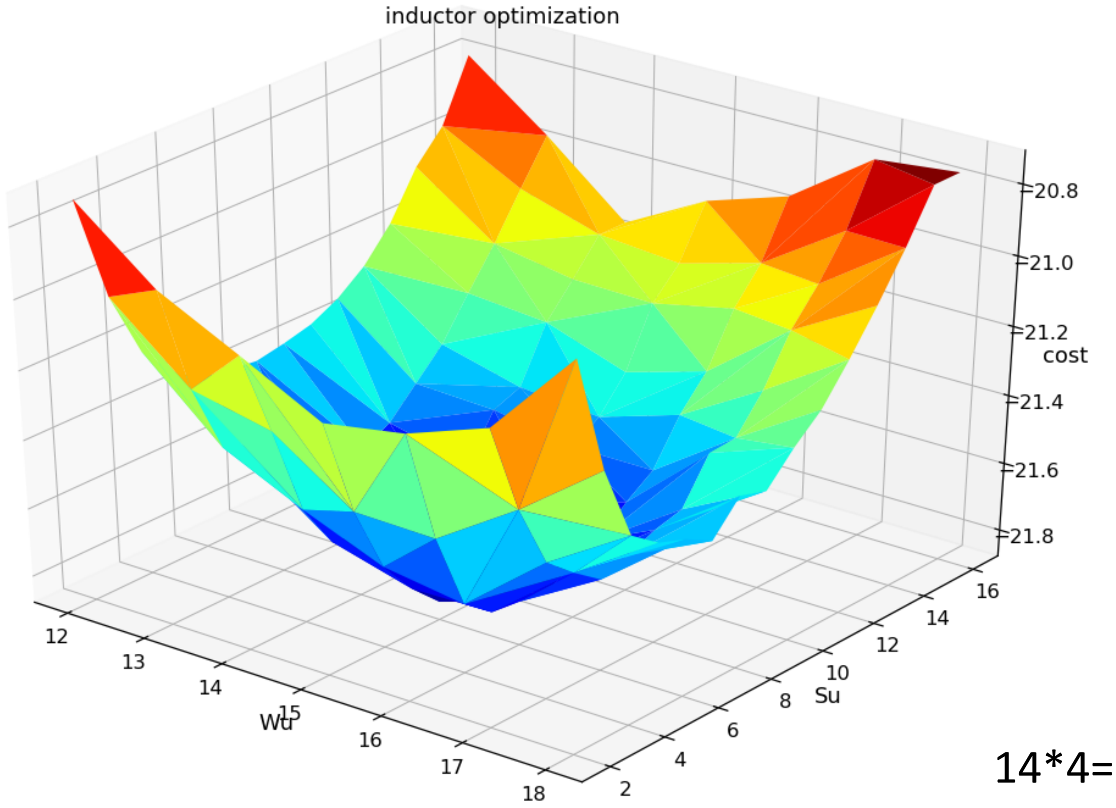






On-chip inductor design

Optimization problem



14*4=56 sims

For each dout and n

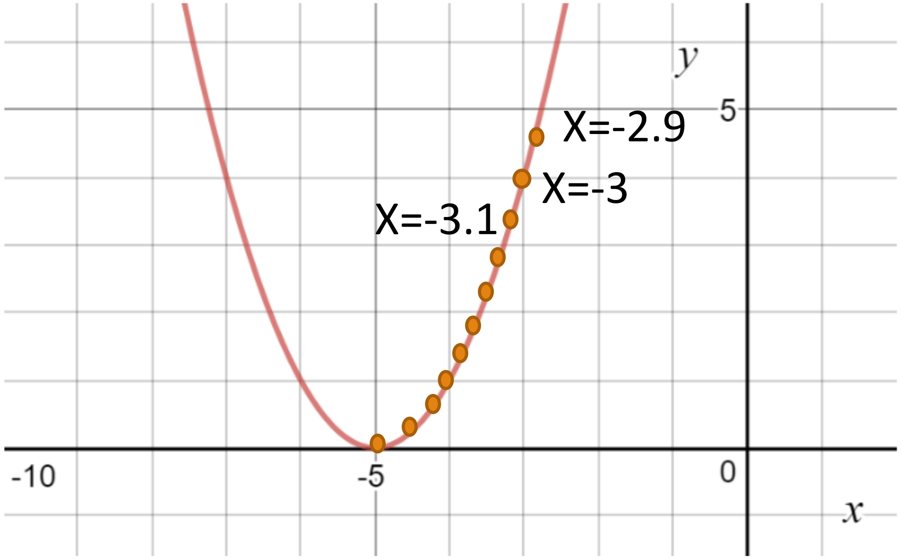
Gradient descent

$J(x)=(x+5)^2$

Start from: x=-3

To find the minimum:

- Increment x by 0.1
- Calculate the gradient
- Increment x by -learnrate*gradient.
  - Learnrate=0.2
- Repeat.



$$Gradient: \frac{dJ(x)}{dx} = \frac{J(x+\Delta x)-J(x)}{\Delta x}$$

2D problem

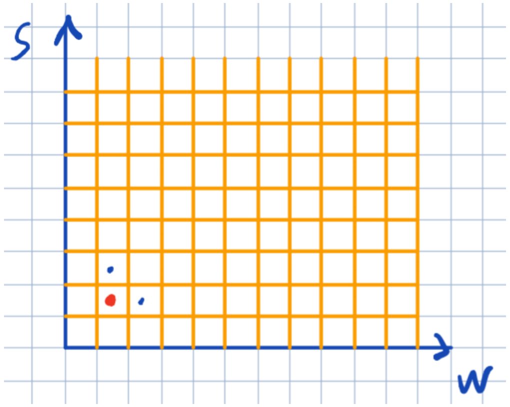
$$\boxed{x_0, y_0}$$

$$\boxed{\frac{\partial J}{\partial x}, \frac{\partial J}{\partial y}}$$

$$\boxed{\begin{aligned} x_{i+1} &= x_i - \frac{\partial J}{\partial x}\Big|_{x=x_i} \cdot \alpha \\ y_{i+1} &= y_i - \frac{\partial J}{\partial y}\Big|_{y=y_i} \cdot \alpha \end{aligned}}$$

Implementation:

Define boundaries/constraints
Set initial guess

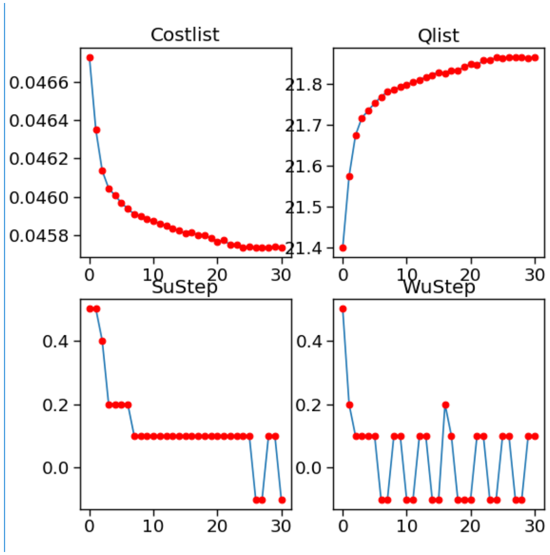


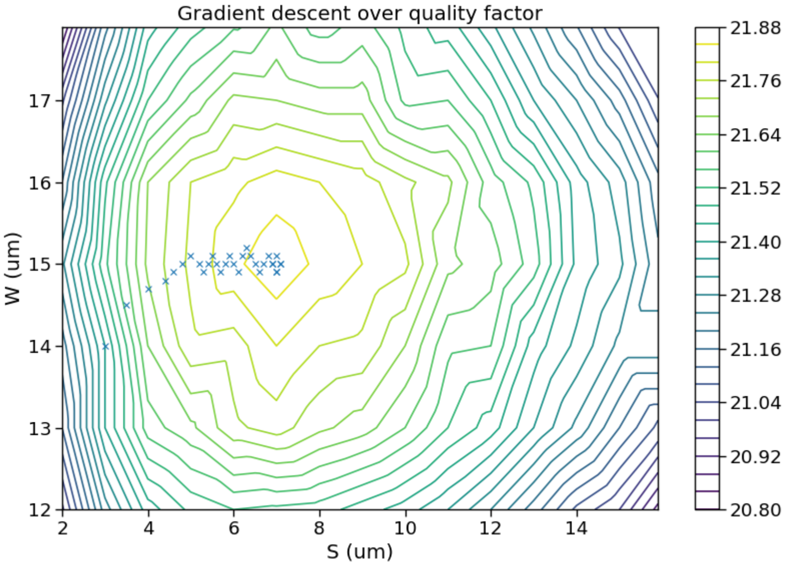
```
L0,Q0=getLandQ(Wu,Su)

Lw,Qw=getLandQ(Wu+step_Wu,Su)
Ls,Qs=getLandQ(Wu,Su+step_Su)

costw=costfunc(Lw,Qw)
costs=costfunc(Ls,Qs)

step_Wu=-learnRate*gradients(cost0,costw,step_Wu)
step_Su=-learnRate*gradients(cost0,costs,step_Su)
```

Advance steps
Repeat

Simulation of the simulation

Testing the algorithm on simulated data.





Thank you!

QUESTIONS?