UNIVERSITÉ DE LORRAINE

Institut des sciences du Digital Management & Cognition
COMPOSANTE DE L'UNIVERSITÉ DE LORRAINE

Programming

8- Summary and Common Libraries

These slides will be available on Arche

By Gaël Guibon, inspired by Mathieu d'Aquin's course

Variables

```
a = 12 # variable assignment
type(a) # type of a variable
b = a * 2 # arithmetic operations on variables
a == b # comparison of variable values
a is b # comparison of variables
s1 = "a string"
s2 = 'another string'
s3 = ''' a multiline
string'''
len(s1) # length of a string
s1[3] # 4th character of s1
s1[3:] # substring after the 4th character
s1[1:-2:-2] # every other character in reverse from the penultimate one to
the second one
```

Conditionals / branching

condition

code for when condition is True

```
if a == b: print("one line if")
else: print("and one line else")
```

code for when condition is False

condition

```
if (a > b or c != d) and a not is d:
    print("yeah, first try")
    print("let's hope it gets better")
elif a is d:
    print("just in case")
else:
    print("OK then")
```

indentation after ":"

code for when condition is True

condition

code for when previous conditions are False but this one is True

indentation after ":"

indentation after ":"

code for when all previous conditions are False

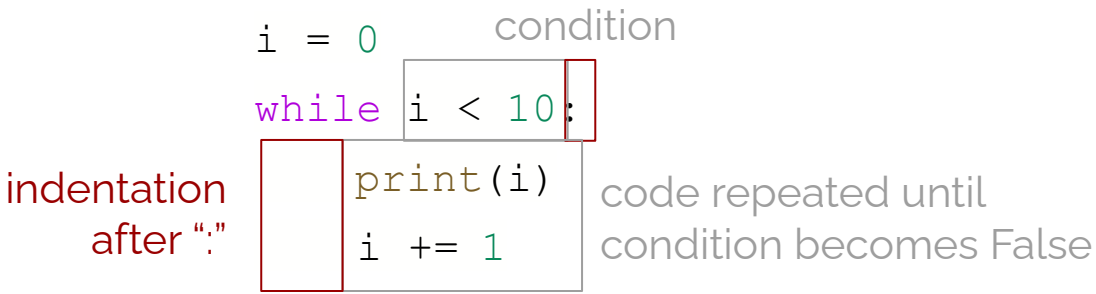
Lists

```
l = [1, 2, "bob", 3.5, True, [6,7]] # a list
len(l) # length of the list
l[2] # third element in list
l[3:] # sublist from fourth element
l[:4] # sublist from fifth element
l[2:5] # sublist from third to fifth elements
l[-1:0;-1] # list in reverse from last to first elements
l[:] # copy of the list (like l.copy())
l.append("alice") # add element to list
l.extend([9,10]) # concatenate lists
3 in l # is the value 3 in l?
```

Dictionaries / tuples

```
d = {"a": 12, "b": [1,2], 34.4: "bob", 5: {"f": 6, "z": 7}} # a dictionary
len(d) # number of attributes in d
d["b"] # value of attribute "b" in d
d[34.4] # value of attribute 34.4 in d
5 in d # is the attribute 5 in d?
d["c"] = 12 # assigning a value to an attribute of d

t = ("b", 14, "jeff", 3.5, [1,2]) # a tuple
len(t) # lenght of a tuble
t[4] # fifth element in t
t[4:2:-1] # sub-tuple from fifth to third elements in reverse
"jeff" in t # is the value "jeff" in t?
```

while loop


i = 0
condition
while i < 10:
print(i)
i += 1
indentation
after ":"
code repeated until
condition becomes False

for loop 1

variable that will take successive values of the range

range

```
for i in range(0,10,1):
    print(i)
    print("carry on")
```

indentation after ":"

code repeated until i has reached the end of the range

for loop 2 (on list)

```
l= [12,13,14]
```

variable that will take successive values in the list

list

```
for x in l:
    print(x)
    print("let's go")
```

indentation after ":"

code repeated until we reach the end of the list

for loop 3 (on list 2)

```
l= [12,13,14]

for i,x in enumerate(l):
    print(i,x)
    print("let's go")
```

tuple that will take successive indices and values in the list

enumerate function on list

indentation after ":"

code repeated until we reach the end of the list

for loop 4 (on dict)

```
d= {"a":12, "b": 13, "c": 14}
```

variable that will take successive attribute names in d

dictionary

```
for k in d:
    print(k,d[k])
    print("let's go")
```

indentation after ":"

code repeated until we reach the end of the list of attributes of the dict

for loop 5 (on dict 2)

```
d= {"a":12, "b": 13, "c": 14}
```

tuples that will take successive attribute names and values in d

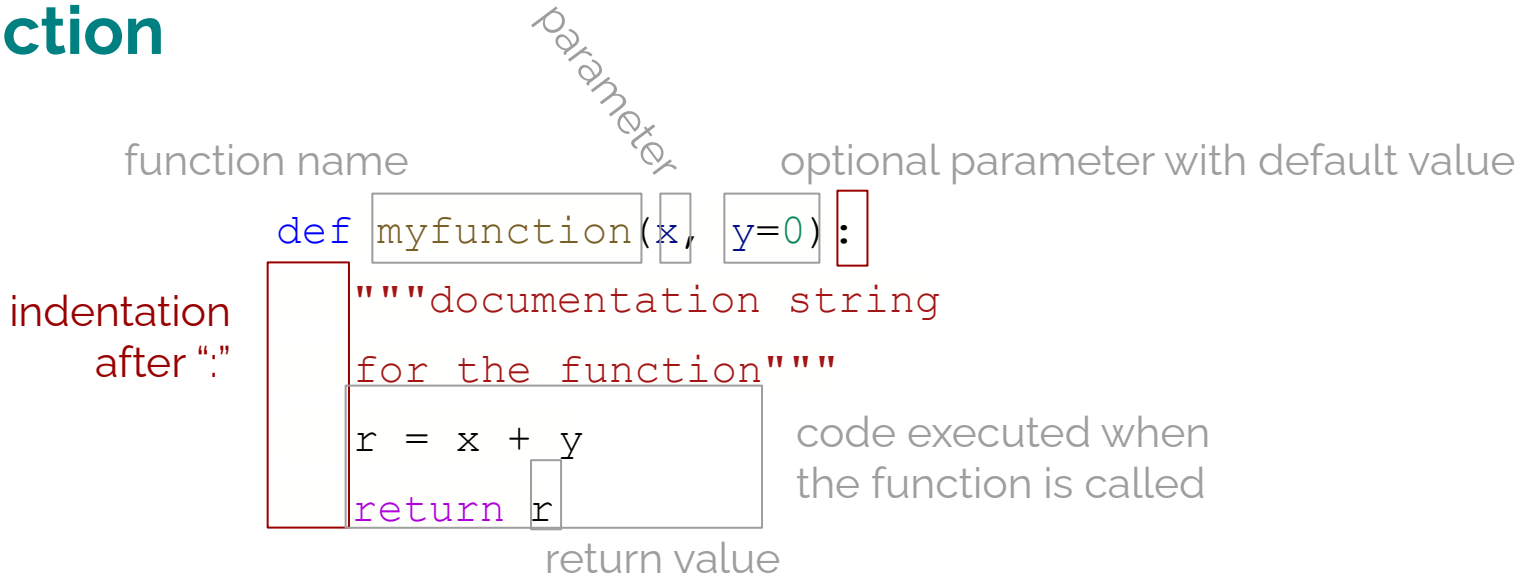
items method on dictionaries

```
for k,v in d.items():
    print(k,v)
    print("let's go")
```

indentation after ":"

code repeated until we reach the end of the list of attributes/values of the dict

function




```
a = myfunction(3)     # call to the function
b = myfunction(a,2)   # call with option parameter set
```

recursive function

```
def rmul(x,y):
    if y == 0: return 0  # base case
    return x+rmul(x,y-1) # recursive step
```

Handling errors

```
import random

try:    # below is the code that might raise an error
    r = random.randrange(0,2)
    a = 10/r
    print(a)
except ZeroDivisionError: # code when error is raised
    print("oops")
```

raise an error

```
def rmul(x,y):
    if type(y) != int: raise TypeError("y should be an integer")
    if y < 0: raise ValueError("y should be greater than 0")

    if y == 0: return 0  # base case
    return x+rmul(x,y-1) # recursive step
```

classes

```
class A:      # class called A
    a = 12    # class attribute

class B(A):  # class B subclass of A
    def __init__(self, v):  # constructor
        self.a = v          # setting instance attribute value
    def addA(x):            # method of the class
        return self.a + x

i = B(7)       # instance of the class
print(i.a)     # print attribute a of i
i.a = 8        # change attribute a of i
y = i.addA(x) # call method of class B on i
```

classes

indentation after ":"

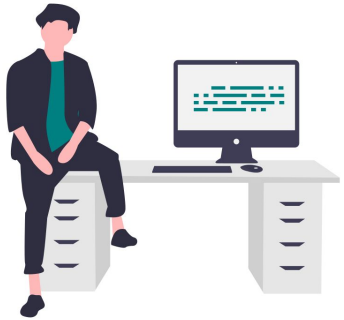
```
class A:      # class called A
    a = 12    # class attribute

class B(A):  # class B subclass of A
    def __init__(self, v):  # constructor
        self.a = v          # setting instance attribute value
    def addA(x):            # method of the class
        return self.a + x


i = B(7)       # instance of the class
print(i.a)     # print attribute a of i
i.a = 8        # change attribute a of i
y = i.addA(x) # call method of class B on i
```

classes

indentation after ":"

```
class A:      # class called A
    a = 12    # class attribute

class B(A):  # class B subclass of A
    def __init__(self, v):  # constructor
        self.a = v          # setting instance attribute value
    def addA(x):            # method of the class
        return self.a + x


i = B(7)       # instance of the class
print(i.a)     # print attribute a of i
i.a = 8        # change attribute a of i
y = i.addA(x) # call method of class B on i
```

Common Modules & Packages



Standard and Installed Modules

You can get the list of modules installed in your environment by using the command (it might take a bit of time)

```
help("modules")
```

```
Cython                colorcet            kapre                 readline
IPython               colorlover          keras                 regex
OpenGL                colorsys            keras_preprocessing   reprlib
PIL                   community           keyword               requests
ScreenResolution      compileall          kiwisolver            requests_oauthlib
__future__            concurrent          korean_lunar_calendar resampy
_abc                  configparser        lib2to3               resource
_ast                  contextlib          libfuturize           retrying
_asyncio              contextlib2         libpasteurize         rlcompleter
_bisect               contextvars         librosa               rmagic
_blake2               convertdate         lightgbm              rpy2
...                   ...                 ...                   ...
```

Standard and Installed Modules

You can also find a list of modules that are included with python (3) at https://docs.python.org/3/py-modindex.html

i.e. the ones that you don't need to install (e.g. using pip)

Examples

audioop module to manipulate audio data

copy functions to make copies (e.g. deepcopy)

csv for CSV files

datetime for objects (classes) representing dates and times

json module to parse and write data in the JSON format

math math functions (abs, round, etc.)

os functions related to interacting with the os

sys functions related to interacting with the python system

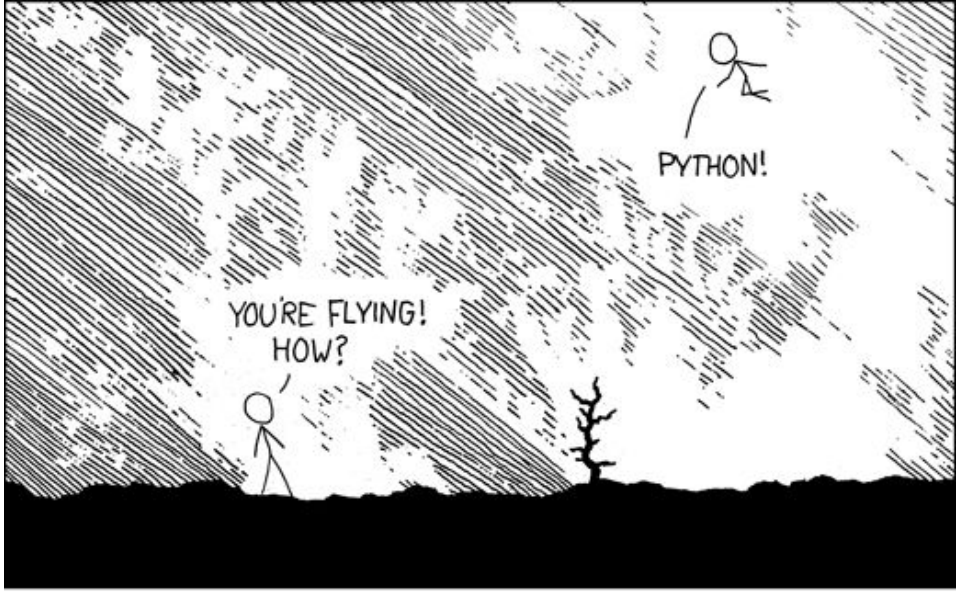
zipfile read and write zip archive

Other Common Installable Modules

Here, we will see some examples of commonly used modules and packages for:

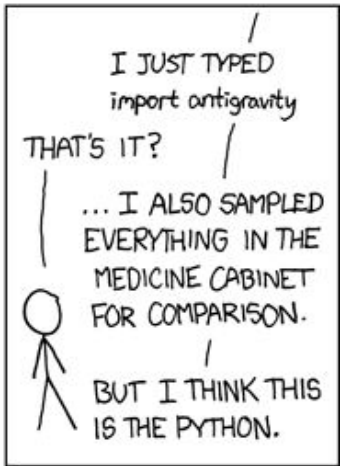
- Interaction
- Data analysis and databases
- Parallel computing
- Web

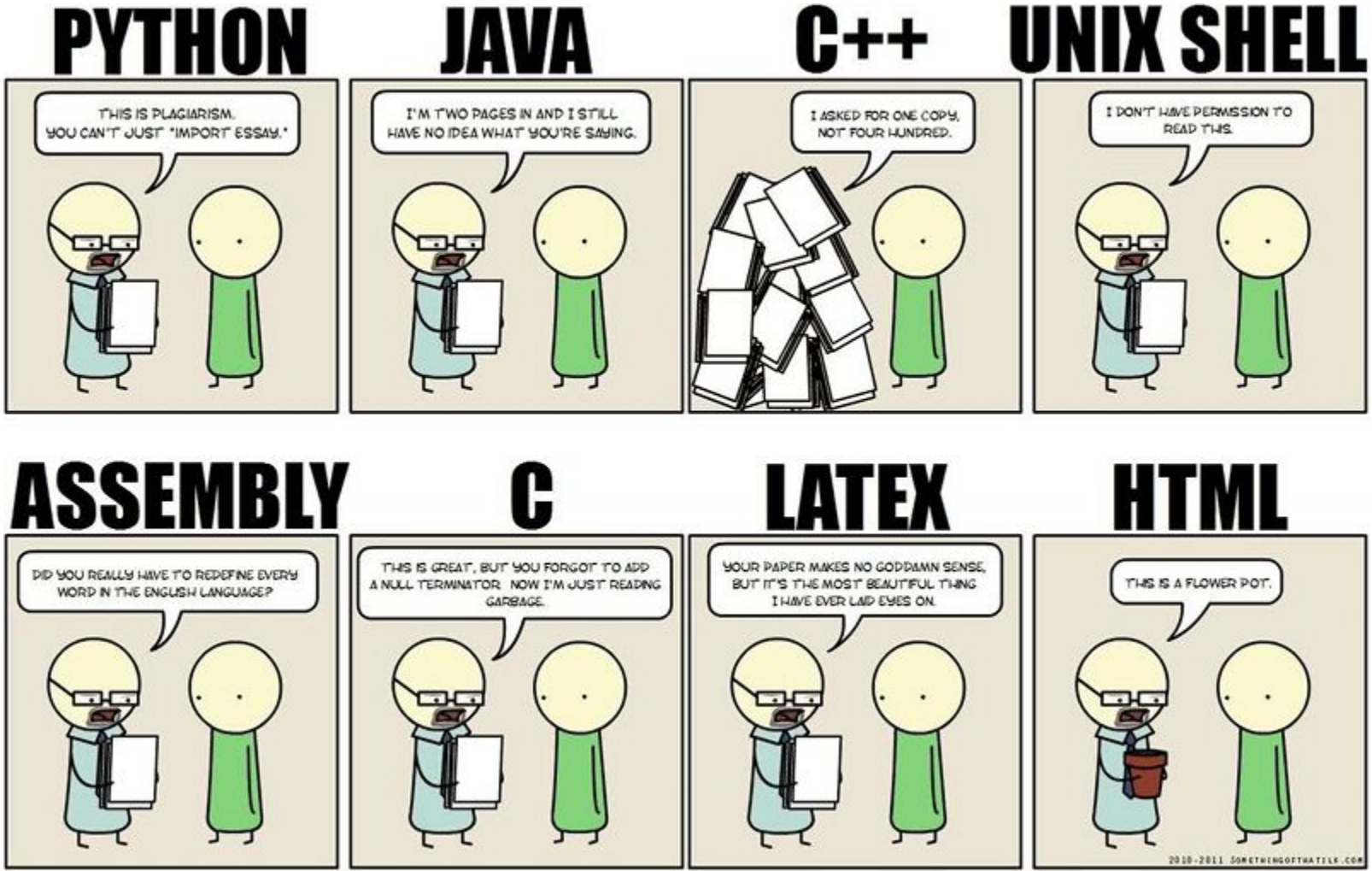
But there are many others that can do all sorts of things…


PYTHON!
YOU'RE FLYING! HOW?


I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"


I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?


I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.


PYTHON
THIS IS PLAGIARISM. YOU CAN'T JUST "IMPORT ESSAY."
JAVA
I'M TWO PAGES IN AND I STILL HAVE NO IDEA WHAT YOU'RE SAYING.
C++
I ASKED FOR ONE COPY, NOT FOUR HUNDRED.
UNIX SHELL
I DON'T HAVE PERMISSION TO READ THIS.
ASSEMBLY
DID YOU REALLY HAVE TO REDEFINE EVERY WORD IN THE ENGLISH LANGUAGE?
C
THIS IS GREAT, BUT YOU FORGOT TO ADD A NULL TERMINATOR. NOW I'M JUST READING GARBAGE.
LATEX
YOUR PAPER MAKES NO GODDAMN SENSE, BUT IT'S THE MOST BEAUTIFUL THING I HAVE EVER LAID EYES ON.
HTML
THIS IS A FLOWER POT.
2010-2011 SOMETHINGOFTHATILK.COM


After I get all the infinity stones, one snap will kill half the universe
Srsly? All I need is 2 lines of code
kill.py
from universe import kill
universe.kill(universe.population/2)
coderforevers

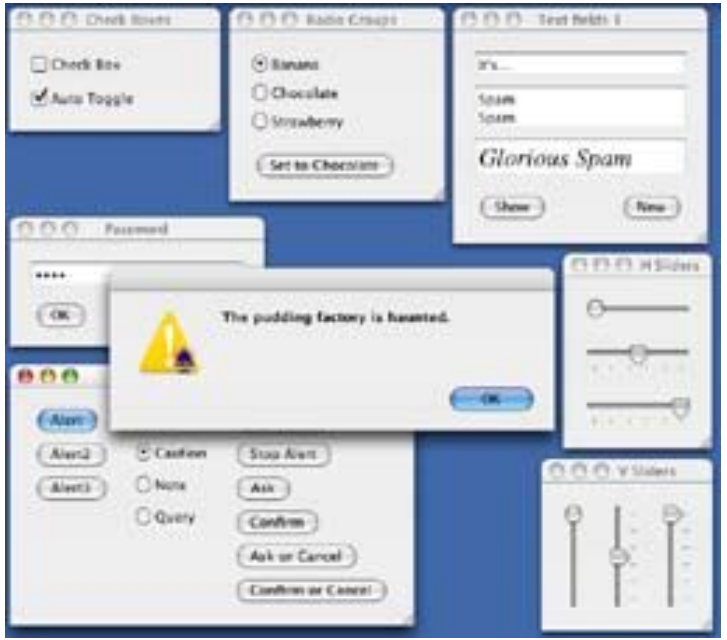
ooops!


After I get all the infinity stones, one snap will kill half the universe
Srsly? All I need is 2 lines of code
kill.py
from universe import kill
universe.kill(universe.population/2)
coderforevers

Interaction

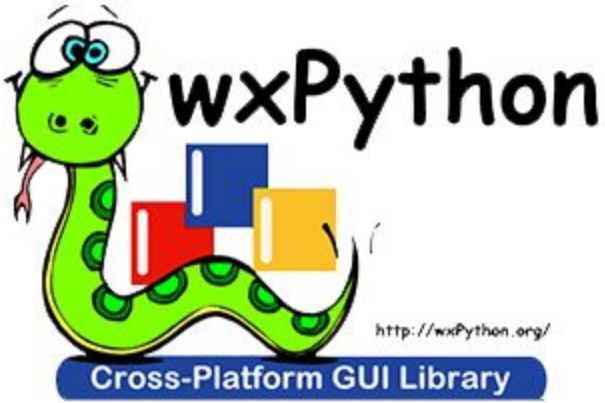
PyGUI

A very simple library for creating interfaces.



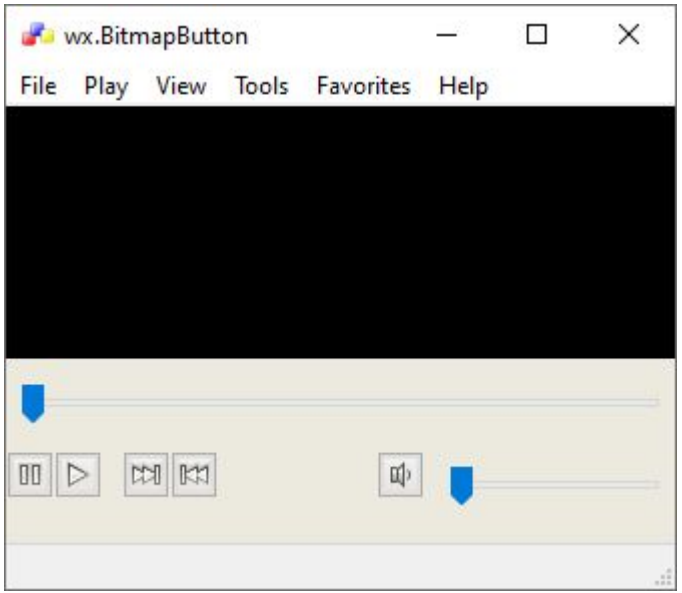
wxPython

Also a library to create GUIs in python, but a bit more advanced and with other libraries (e.g. wax) developed on top of it to make it simpler.



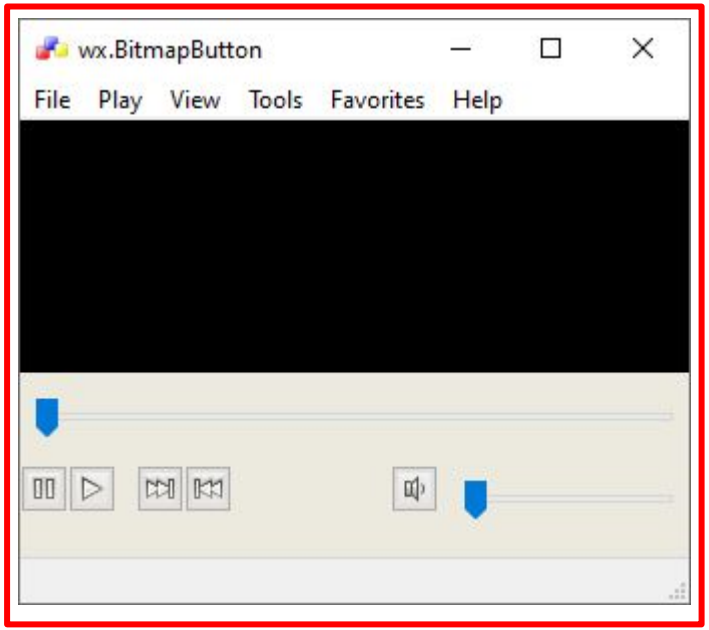
wxPython example

```
import wx
class MyFrame(wx.Frame):
  def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id,
title, size=(350, 300))
        ...
class MyApp(wx.App):
  def OnInit(self):
        frame = MyFrame(None, -1,
'wx.BitmapButton')
        frame.Show(True)
        self.SetTopWindow(frame)
        return True
app = MyApp(0)
app.MainLoop()
```



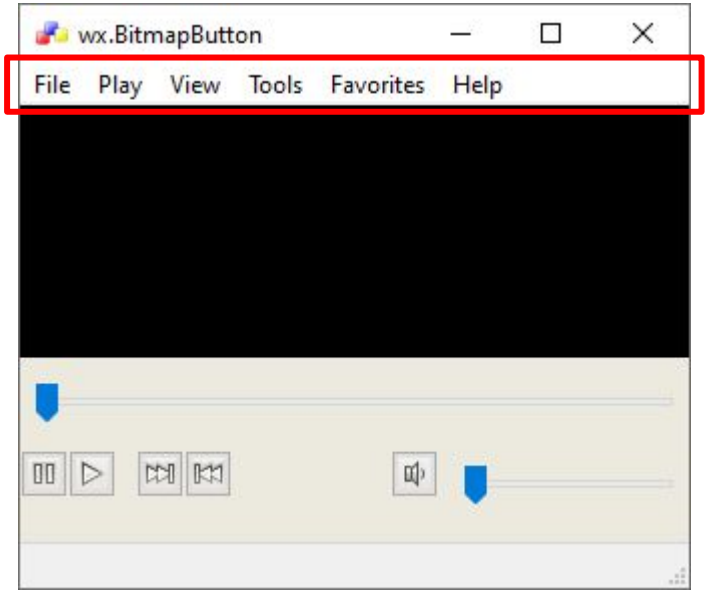
wxPython example

```
import wx
class MyFrame(wx.Frame):
  def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id,
title, size=(350, 300))
        ...
class MyApp(wx.App):
  def OnInit(self):
        frame = MyFrame(None, -1,
'wx.BitmapButton')
        frame.Show(True)
        self.SetTopWindow(frame)
        return True
app = MyApp(0)
app.MainLoop()
```



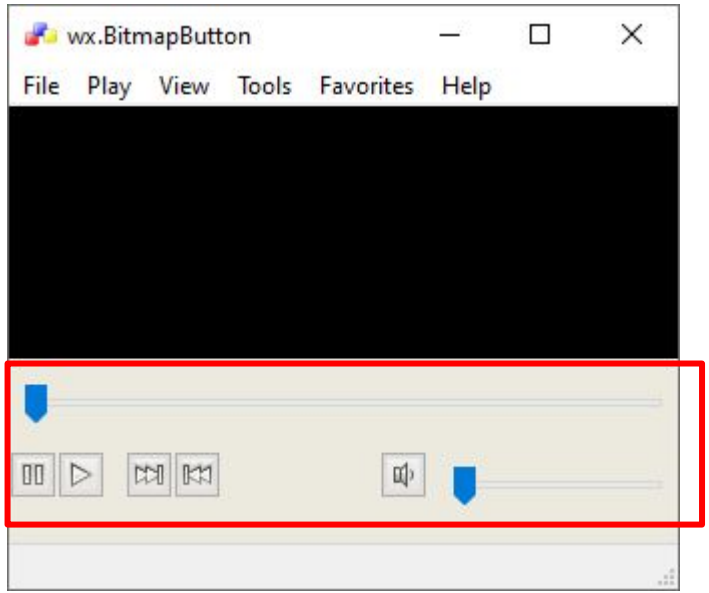
wxPython example

```
menubar = wx.MenuBar()
file = wx.Menu()
play = wx.Menu()
view = wx.Menu()
tools = wx.Menu()
favorites = wx.Menu()
help = wx.Menu()

file.Append(101, '&quit', 'Quit
application')

menubar.Append(file, '&File')
menubar.Append(play, '&Play')
menubar.Append(view, '&View')
menubar.Append(tools, '&Tools')
menubar.Append(favorites, 'F&avorites')
```



wxPython example

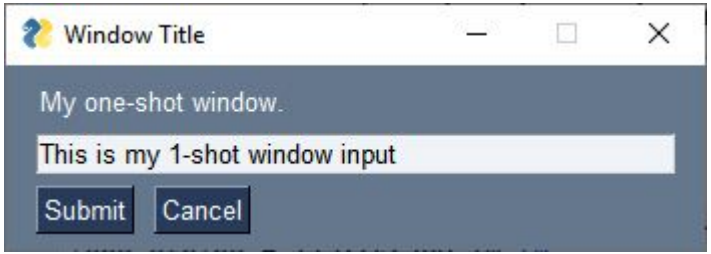
```
slider1 = wx.Slider(pnl2, -1, 0, 0, 1000)
pause = wx.BitmapButton(pnl2, -1,
wx.Bitmap('./bitmaps/stock_media-pause.png' )
)
play  = wx.BitmapButton(pnl2, -1,
wx.Bitmap('./bitmaps/stock_media-play.png' ))
next  = wx.BitmapButton(pnl2, -1,
wx.Bitmap('./bitmaps/stock_media-next.png' ))
prev  = wx.BitmapButton(pnl2, -1,
wx.Bitmap('./bitmaps/stock_media-prev.png' ))
volume = wx.BitmapButton(pnl2, -1,
wx.Bitmap('./bitmaps/volume.png' ))
slider2 = wx.Slider(pnl2, -1, 0, 0, 100,
size=(120, -1))
```

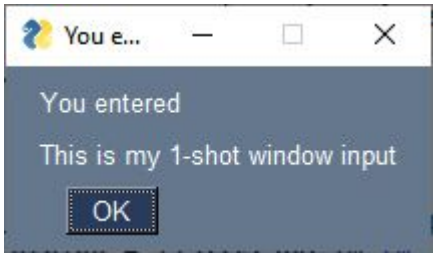


https://wiki.wxpython.org/How%20to%20create%20a%20bitmap%20button%20(Phoenix)

PySimpleGUI

```
import PySimpleGUI as sg

layout = [[sg.Text('My one-shot window.')],
          [sg.InputText()],
          [sg.Submit(), sg.Cancel()]]

window = sg.Window('Window Title', layout)

event, values = window.read()
window.close()

text_input = values[0]
sg.popup('You entered', text_input)
```

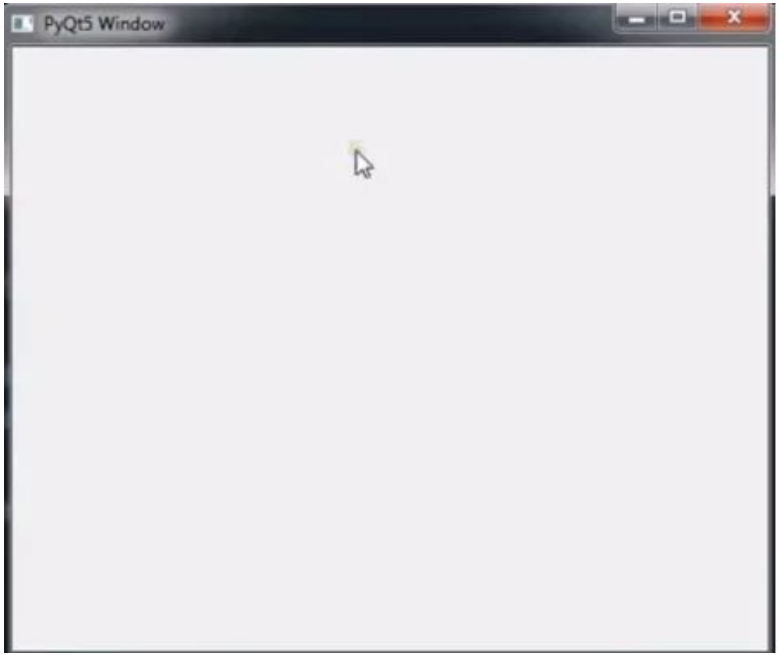




https://wiki.wxpython.org/How%20to%20create%20a%20bitmap%20button%20(Phoenix)

PyQT5

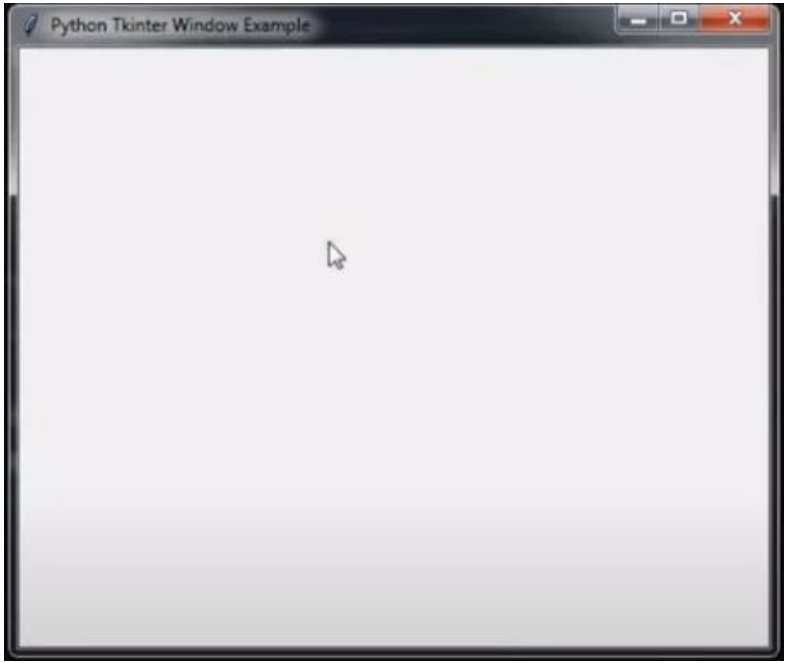
```
from PyQt5.QtWidgets import QApplication,
QMainWindow
import sys
class Window(QMainWindow):
  def __init__(self):
      super().__init__()
      self.setGeometry(300, 300, 600, 400)
      self.setWindowTitle("PyQt5 window")
      self.show()
app = QApplication(sys.argv)
window = Window()
sys.exit(app.exec_())
```



tkinter

```
from tkinter import *
class Root(Tk):
    def __init__(self):
        super(Root, self).__init__()
        self.title("Python Tkinter")
        self.minsize(500,400)
root = Root()
root.mainloop()
```

And many others…

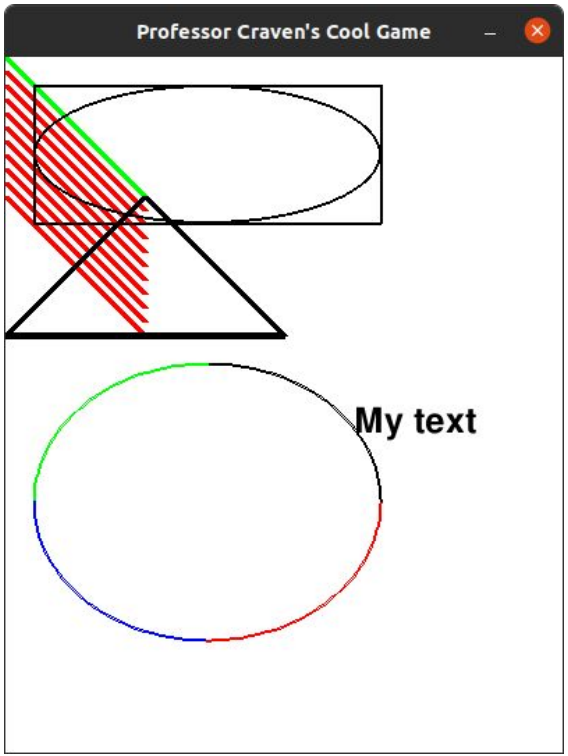


Pillow (image manipulation)

```
from PIL import Image, ImageFilter
img = Image.open("../../_static/pillow.png")
img = img.filter(ImageFilter.BLUR) \
        .transpose(Image.FLIP_TOP_BOTTOM) \
        .transpose(Image.FLIP_LEFT_RIGHT)
width, height = img.size
WHITE_THRESHOLD = 250
PURPLE = (155, 89, 182)
BLUE = (52, 152, 219)
for x in range(width):
    for y in range(height):
        pixel = img.getpixel((x, y))
        if all(channel > WHITE_THRESHOLD for channel in pixel):
            if x + y <= width:
                img.putpixel((x, y), PURPLE)
            else:
                img.putpixel((x, y), BLUE)
img.show()
```

PyGame

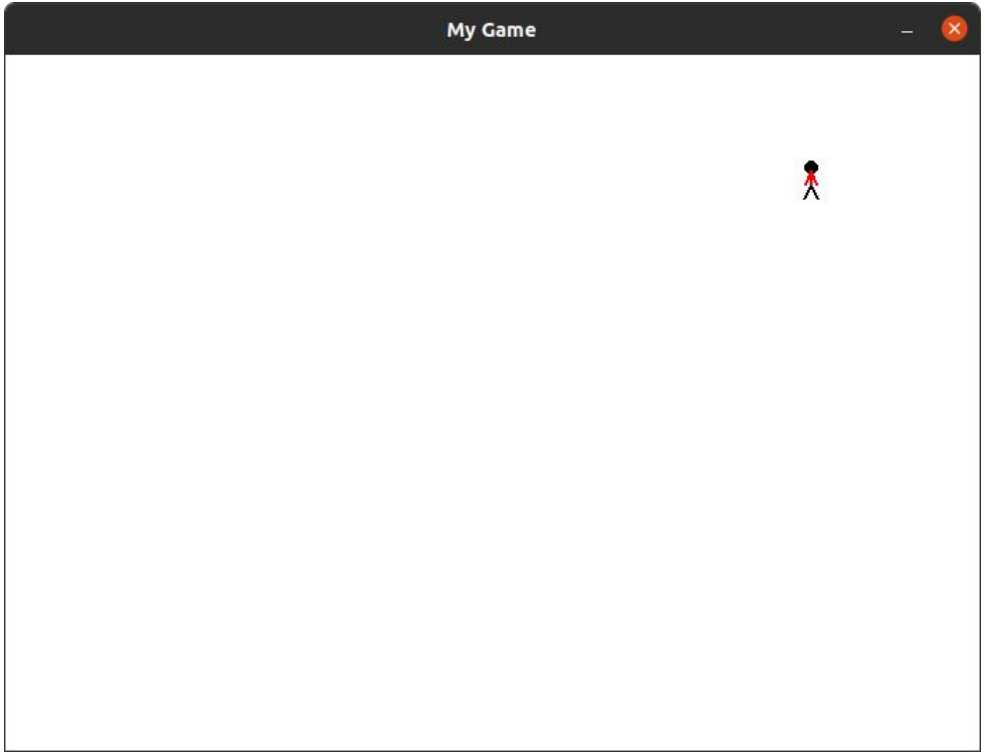
Drawing



Animating

PyGame

Inputs



PyGame

Music

```
pygame.mixer.music.load('MIT_Concert_Choir_O_Fortuna.ogg')
pygame.mixer.music.set_endevent(pygame.constants.USEREVENT)
pygame.mixer.music.play()
```

Data Analysis

numpy

numpy is a library for **scientific computing** which includes types and operations for more precise and faster number manipulation, arrays, etc. It is **used in many other libraries** mentioned in the next few slides.

```
>>> a = np.array([20, 30, 40, 50])
>>> b = np.arange(4)
>>> b
array([0, 1, 2, 3])
>>> c = a - b
>>> c
array([20, 29, 38, 47])
>>> b**2
array([0, 1, 4, 9])
>>> 10 * np.sin(a)
array([ 9.12945251, -9.88031624,  7.4511316 , -2.62374854])
>>> a < 35
array([ True,  True, False, False])
```

pandas

pandas is a popular library for manipulating DataFrames (**data tables**) and data series, especially for **statistical analysis**.

```
import pandas as pd
df = pd.read_csv("mk_bodies.csv", index_col=0)
print(df.index)
df.describe()
```

```
_______________
Index(['Standard Kart', 'Pipe Frame', 'Mach 8', 'Cat Cruiser', 'Steel Driver',
       'Circuit Special', 'Tri-Speeder', 'Badwagon', 'Prancer', 'Biddybuggy',
       'Landship', 'Sneeker', 'Sports Coupe', 'Gold Standard', 'Mercedes GLA',
       'Mercedes Silver Arrow', 'Mercedes 300 SL Roadster', 'Blue Falcon',
       'Tanooki Kart', 'B Dasher', 'Streetle', 'P-Wing', 'Koopa Clown',
       'Standard Bike', 'Comet', 'Sport Bike', 'The Duke', 'Flame Rider',
       'Varmint', 'Mr. Scooty', 'Jet Bike', 'Yoshi Bike', 'Master Cycle',
       'City Tripper', 'Standard ATV', 'Wild Wiggler', 'Teddy Buggy',
       'Bone Rattler', 'Inkstriker', 'Splat Buggy'],
      dtype='object', name='Vehicle')
```

pandas

pandas is a popular library for manipulating DataFrames (**data tables**) and data series, especially for **statistical analysis**.

```
import pandas as pd
df = pd.read_csv("mk_bodies.csv", index_col=0)
print(df.index)
df.describe()

_____________
Index(['Standard Kart
       'Circuit Speci
       'Landship', 'S
       'Mercedes Silv
       'Tanooki Kart'
       'Standard Bike
       'Varmint', 'Mr
       'City Tripper'
       'Bone Rattler'
      dtype='object',
```

	Speed	Acceleration	Weight	Handling	Traction	Mini Turbo
count	40.000000	40.000000	40.000000	40.000000	40.000000	40.000000
mean	-0.018750	-0.168750	-0.012500	-0.062500	0.068750	-0.006250
std	0.364215	0.516979	0.324975	0.382929	0.371188	0.504745
min	-0.750000	-1.000000	-0.500000	-0.750000	-0.750000	-1.000000
25%	-0.250000	-0.500000	-0.250000	-0.500000	-0.062500	-0.250000
50%	0.000000	-0.250000	0.000000	0.000000	0.000000	0.000000
75%	0.250000	0.250000	0.250000	0.250000	0.312500	0.250000
max	0.500000	0.750000	0.500000	0.500000	0.750000	1.000000

SQLite

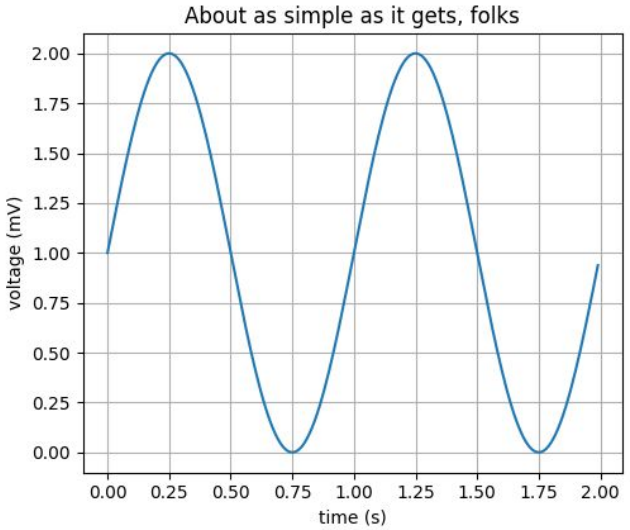
SQLite3 is a simple database technology which makes it easy to **create, update and query a database stored in a simple file**.

```
import sqlite3

conn = sqlite3.connect('test.db')
cursor = conn.execute("SELECT id, name, address, salary from COMPANY" )
for row in cursor:
   print("ID = ", row[0])
   print("NAME = ", row[1])
   print("ADDRESS = ", row[2])
   print("SALARY = ", row[3], "\n")
conn.close()
```

matplotlib

matplotlib is the most popular library to **create graphs** in python.

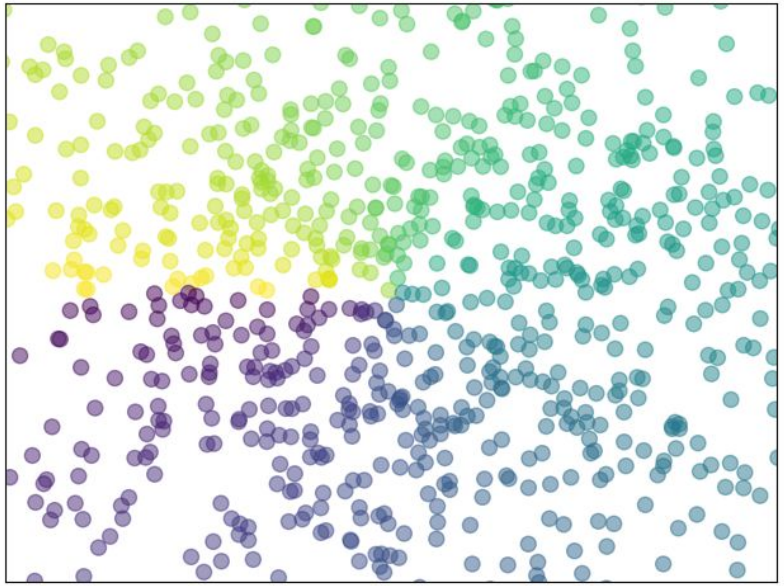
```
import matplotlib.pyplot as plt
import numpy as np


t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2*np.pi*t)
plt.plot(t, s)


plt.xlabel('time (s)')
plt.ylabel('voltage (mV)')
plt.title('About as simple as it gets, folks' )
plt.grid(True)
plt.savefig("test.png")
plt.show()
```



matplotlib

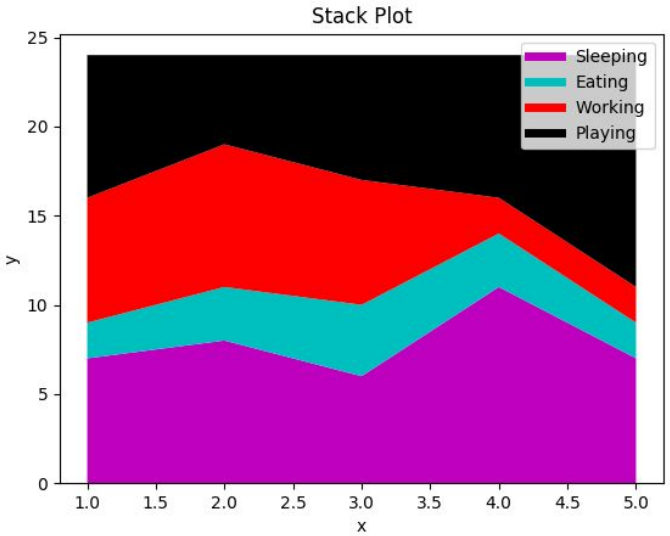
matplotlib is the most popular library to create graphs in python.

```
import numpy as np
import matplotlib.pyplot as plt

n = 1024
X = np.random.normal(0, 1, n)
Y = np.random.normal(0, 1, n)
T = np.arctan2(Y, X)

plt.axes([0.025, 0.025, 0.95, 0.95])
plt.scatter(X, Y, s=75, c=T, alpha=.5)

plt.xlim(-1.5, 1.5)
plt.xticks([])
plt.ylim(-1.5, 1.5)
plt.yticks([])

plt.show()
```



matplotlib

matplotlib is the most popular library to create gra

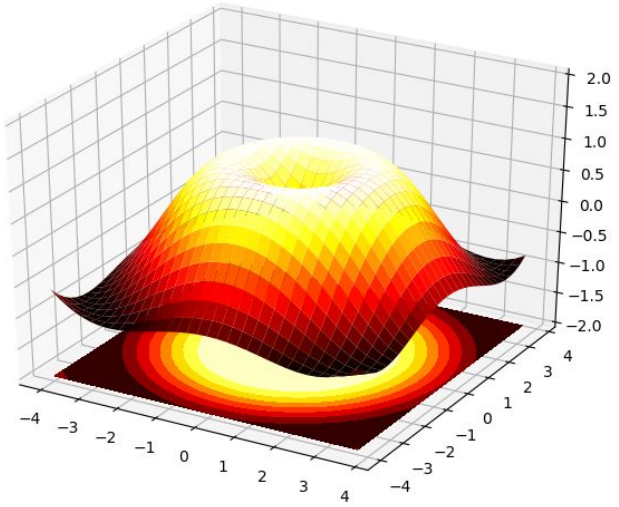



```
import matplotlib.pyplot as plt
days = [1,2,3,4,5]
sleeping =[7,8,6,11,7]
eating = [2,3,4,3,2]
working =[7,8,7,2,2]
playing = [8,5,7,8,13]
plt.plot([],[],color='m', label='Sleeping', linewidth=5)
plt.plot([],[],color='c', label='Eating', linewidth=5)
plt.plot([],[],color='r', label='Working', linewidth=5)
plt.plot([],[],color='k', label='Playing', linewidth=5)
plt.stackplot(days, sleeping,eating,working,playing, colors=['m','c','r','k'])
plt.xlabel('x')
plt.ylabel('y')
plt.title('Stack Plot')
plt.legend()
plt.show()
```

matplotlib

matplotlib is the most popular library to create graphs in python.

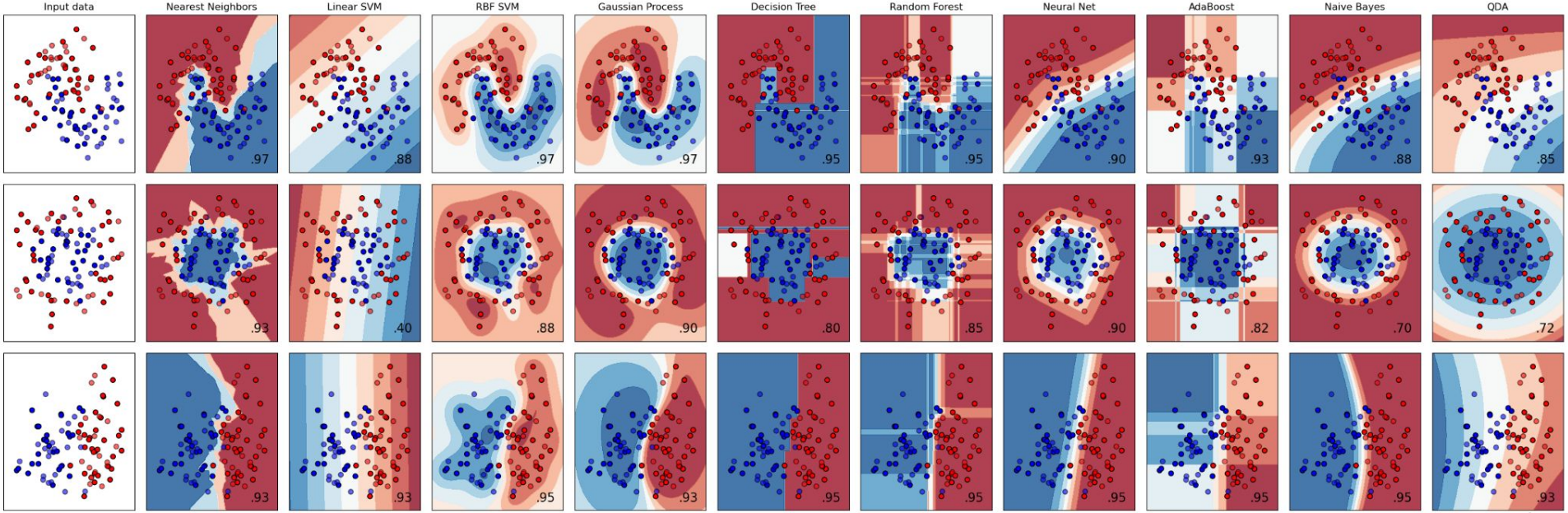
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = Axes3D(fig)
X = np.arange(-4, 4, 0.25)
Y = np.arange(-4, 4, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X ** 2 + Y ** 2)
Z = np.sin(R)

ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=plt.cm.hot)
ax.contourf(X, Y, Z, zdir='z', offset=-2, cmap=plt.cm.hot)
ax.set_zlim(-2, 2)

plt.show()
```



Machine Learning

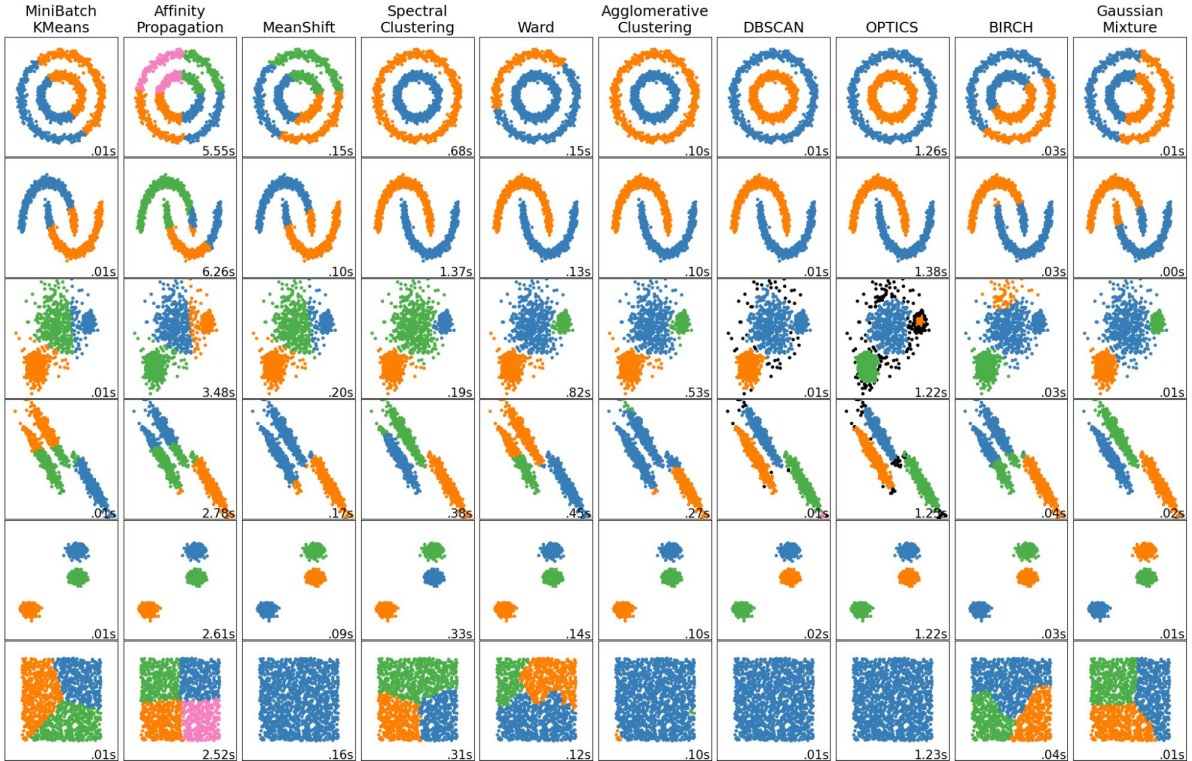
sklearn

SciKit Learn is a library that includes a large number of **machine learning algorithms**, e.g. for classification



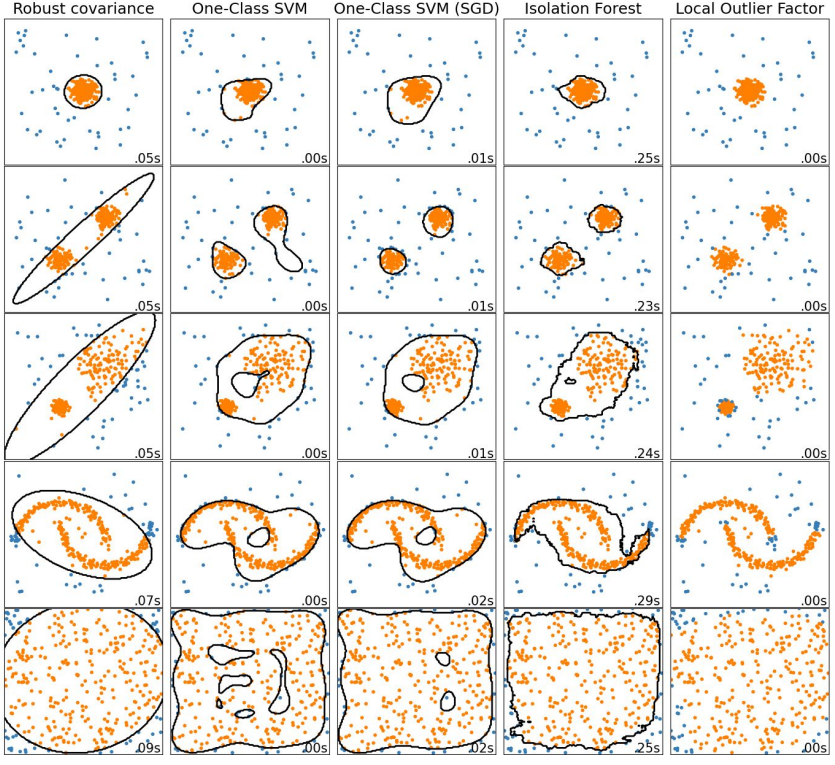
sklearn

SciKit Learn is a library that includes a large number of **machine learning algorithms**, e.g. for clustering



sklearn

SciKit Learn is a library that includes a large number of **machine learning algorithms**, e.g. for anomaly detection



Tensorflow (Google)

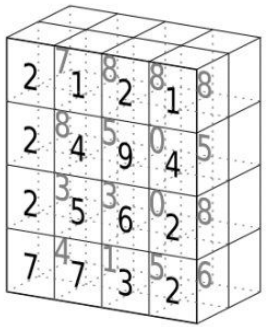
Tensorflow is a library to create, train and apply deep learning models (i.e. large scale artificial neural networks).
Used a lot in **R&D** and **Industry**

't'
'e'
'n'
's'
'o'
'r'

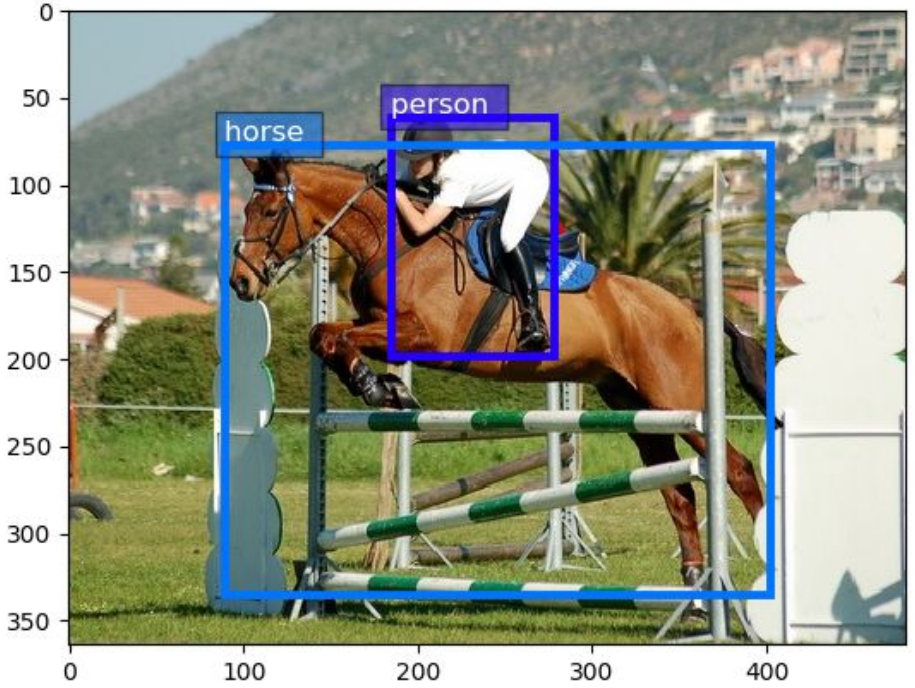
tensor of dimensions [6]
(vector of dimension 6)

3	1	4	1
5	9	2	6
5	3	5	8
9	7	9	3
2	3	8	4
6	2	6	4

tensor of dimensions [6,4]
(matrix 6 by 4)



tensor of dimensions [4,4,2]



Keras

Keras is an overlay framework over Tensorflow.

- Easier access but less customization
- Now integrated into Tensorflow

```
# Input for variable-length sequences of integers
inputs = keras.Input(shape=(None,), dtype="int32")
# Embed each integer in a 128-dimensional vector
x = layers.Embedding(max_features, 128)(inputs)
# Add 2 bidirectional LSTMs
x = layers.Bidirectional(layers.LSTM(64, return_sequences=True))(x)
x = layers.Bidirectional(layers.LSTM(64))(x)
# Add a classifier
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, None)]            0
_________________________________________________________________
embedding (Embedding)        (None, None, 128)         2560000
_________________________________________________________________
bidirectional (Bidirectional (None, None, 128)         98816
_________________________________________________________________
bidirectional_1 (Bidirection (None, 128)               98816
_________________________________________________________________
dense (Dense)                (None, 1)                 129
=================================================================
Total params: 2,757,761
Trainable params: 2,757,761
Non-trainable params: 0
```

PyTorch (Facebook)

PyTorch is a library to create, train and apply deep learning models (i.e. large scale artificial neural networks) **dynamically**

The most popular in **research**! Integrates an **automatic differentiation**!

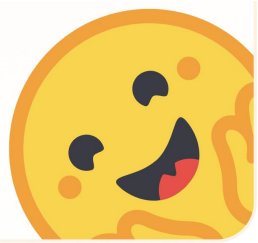
```python
# defining the model
class MinimalExampleModel(torch.nn.Module):
    def __init__(self, D_in, H, D_out):
        """
        In the constructor we instantiate two nn.Linear modules and assign them as
        member variables.
        """
        super(MyModel, self).__init__()
        self.linear1 = torch.nn.Linear(D_in, H, bias=True)
        self.linear2 = torch.nn.Linear(H, D_out, bias=True)

    def forward(self, x):
        """
        In the forward function we accept a Tensor of input data and we must return
        a Tensor of output data. We can use Modules defined in the constructor as
        well as arbitrary operators on Tensors.
        """
        h_relu = torch.relu(self.linear1(x))
        y_pred = self.linear2(h_relu)
        return y_pred
```

Transformers

Transformers is a state-of-the-art Machine Learning (based on PyTorch, Tensorflow)

```
1 from transformers import pipeline
2 classifier = pipeline("sentiment-analysis")
3 classifier("We are very happy to show you the 🤗 Transformers library.")
```

```
[{'label': 'POSITIVE', 'score': 0.9998}]
```



Natural Language Processing

NLTK

The **Natural Language Toolkit** is a powerful library to manipulate text and carry out many of the tasks in Natural language Processing.

```
from nltk.corpus import twitter_samples
from nltk.tag import pos_tag_sents

tweets_tokens = twitter_samples.tokenized('positive_tweets.json')
tweets_tagged = pos_tag_sents(tweets_tokens)
JJ_count = 0
NN_count = 0
for tweet in tweets_tagged:
    for pair in tweet:
        tag = pair[1]
        if tag == 'JJ':
            JJ_count += 1
        elif tag == 'NN':
            NN_count += 1
print('Total number of adjectives = ', JJ_count)
print('Total number of nouns = ', NN_count)
```

Spacy

spaCy is a more recent NLP toolkit to simplify many common tasks.

Edit the code & try spaCy — spaCy v3.4 · Python 3 · via Binder

```
# pip install -U spacy
# python -m spacy download en_core_web_sm
import spacy

# Load English tokenizer, tagger, parser and NER
nlp = spacy.load("en_core_web_sm")

# Process whole documents
text = ("When Sebastian Thrun started working on self-driving cars at "
        "Google in 2007, few people outside of the company took him "
        "seriously. “I can tell you very senior CEOs of major American "
        "car companies would shake my hand and turn away because I wasn’t "
        "worth talking to,” said Thrun, in an interview with Recode earlier "
        "this week.")
doc = nlp(text)

# Analyze syntax
print("Noun phrases:", [chunk.text for chunk in doc.noun_chunks])
print("Verbs:", [token.lemma_ for token in doc if token.pos_ == "VERB"])

# Find named entities, phrases and concepts
for entity in doc.ents:
    print(entity.text, entity.label_)
```

RUN

Features

- Support for **72+ languages**
- **80 trained pipelines** for 24 languages
- Multi-task learning with pretrained **transformers** like BERT
- Pretrained **word vectors**
- State-of-the-art speed
- Production-ready **training system**
- Linguistically-motivated **tokenization**
- Components for **named entity** recognition, part-of-speech tagging, dependency parsing, sentence segmentation, **text classification**, lemmatization, morphological analysis, entity linking and more
- Easily extensible with **custom components** and attributes
- Support for custom models in **PyTorch**, **TensorFlow** and other frameworks
- Built in **visualizers** for syntax and NER
- Easy **model packaging**, deployment and workflow management
- Robust, rigorously evaluated accuracy

Gensim

Gensim is a topic modelling NLP library

```
from gensim import corpora, models, similarities, downloader

# Stream a training corpus directly from S3.
corpus = corpora.MmCorpus("s3://path/to/corpus")

# Train Latent Semantic Indexing with 200D vectors.
lsi = models.LsiModel(corpus, num_topics=200)

# Convert another corpus to the LSI space and index it.
index = similarities.MatrixSimilarity(lsi[another_corpus])

# Compute similarity of a query vs indexed documents.
sims = index[query]
```

(limited) Parallel Computing

Beware of the GIL : Python Global Interpretor Lock

Threading

A thread is a sub-process that can run in parallel to other thread, allowing to achieve some form of parallelisation.

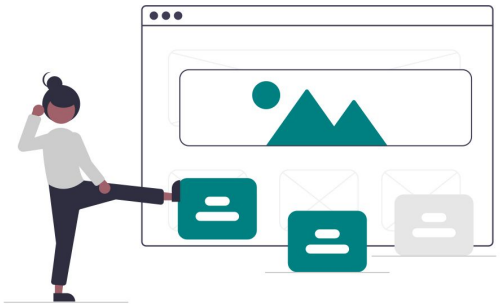
```
import threading

def numbers(start_num):
    for i in range(5):
        print(start_num+i, end=' ')

if __name__ == '__main__':
    t1 = threading.Thread(target=numbers, args=(1,))
    t2 = threading.Thread(target=numbers, args=(10,))
    t1.start()
    t2.start()
    # wait for the processes to finish
    t1.join()
    t2.join()
```

Multiprocessing

multiprocessing is the most commonly used library for running parallel processes in python.

```
import multiprocessing as mp
data = ...
def howmany_within_range(row, minimum, maximum):
    count = 0
    for n in row:
        if minimum <= n <= maximum: count = count +  1
    return count
pool = mp.Pool(mp.cpu_count())
results = [pool.apply(howmany_within_range, args=(row,  4, 8)) for row in
data]
pool.close()
print(results[:10])
```

Web



Requests

requests is a library that implements the HTTP protocol to interact with websites and web APIs

```
import requests

r = requests.get("https://idmc.univ-lorraine.fr/")
print(r.text)
```

```
<a href="#" class="scroll_to_top icon-up" title="Scroll to top"></a>

<div class="custom_html_section">
</div>

<script type="text/javascript">if (typeof AXIOM_UNIVERSITY_STORAGE == 'undefined') var AXIOM_UNIVERSITY_STORAGE
= {};if (AXIOM_UNIVERSITY_STORAGE['theme_font']=='') AXIOM_UNIVERSITY_STORAGE['theme_font'] =
'Roboto';AXIOM_UNIVERSITY_STORAGE['theme_skin_color'] = '';AXIOM_UNIVERSITY_STORAGE['theme_skin_bg_color'] =
'';</script><script type="text/javascript">if (typeof AXIOM_UNIVERSITY_STORAGE == 'undefined') var
AXIOM_UNIVERSITY_STORAGE = {};AXIOM_UNIVERSITY_STORAGE["strings"] = {ajax_error:                    "Invalid
server answer",bookmark_add:          "Add the bookmark",bookmark_added:          "Current page has been
successfully added to the bookmarks. You can see it in the right panel on the tab
&#039;Bookmarks&#039;",bookmark_del:              "Delete this bookmark",bookmark_title:            "Enter
bookmark title",bookmark_exists:      ...
```

BeautifulSoup

BeautifulSoup allows you to parse and extract data from HTML and XML.

```
import requests
from bs4 import BeautifulSoup
r = requests.get("https://idmc.univ-lorraine.fr/")

soup=BeautifulSoup(r.text, 'html.parser')
for link in soup.find_all('a'):
   print(link.get('href'))
```

```
_________________
None
None
https://idmc.univ-lorraine.fr/
#
https://idmc.univ-lorraine.fr/jpo2021/
https://idmc.univ-lorraine.fr/jpo2021/orientation-postbac-espace-lycee/
https://idmc.univ-lorraine.fr/jpo2021/
https://idmc.univ-lorraine.fr/jpo2021/orientation-postbac-espace-etudiants/
https://idmc.univ-lorraine.fr/software-showroom/
https://idmc.univ-lorraine.fr/jpo2021/orientation-postbac-espace-parents/
https://idmc.univ-lorraine.fr/jpo2021/espace-salarie-formation-continue/
https://idmc.univ-lorraine.fr/idmc/
https://idmc.univ-lorraine.fr/idmc/
https://idmc.univ-lorraine.fr/equipe-de-lidmc/
```

Scrapy

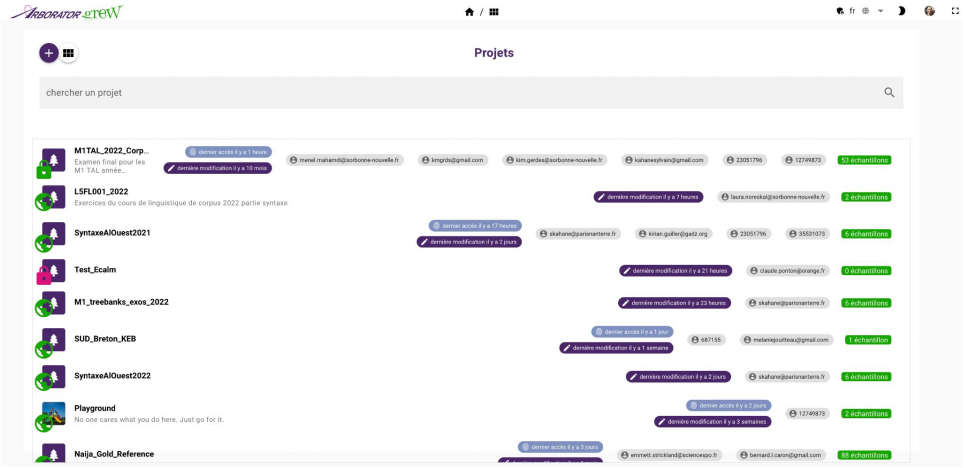
Scrapy does similar things to BeautifulSoup, but can also create web crawlers/spiders

```
import scrapy

class BlogSpider(scrapy.Spider):
    name = 'blogspider'
    start_urls = ['https://www.zyte.com/blog/']

    def parse(self, response):
        for title in response.css('.oxy-post-title'):
            yield {'title': title.css('::text').get()}

        for next_page in response.css('a.next'):
            yield response.follow(next_page, self.parse)
```

Flask

Flask is a complete framework to create web applications / web APIs.

```
from flask import Flask, render_template

app = Flask(__name__)

message = "This is a message"

@app.root('/message/show')
def show_message():
 return f"<p>{message}</p>"

@app.root('/message/new/<newmessage>', methods=["POST"])
def new_message(newmessage):
  message = newmessage
  return "<p>message updated<p>"

@app.route('/')
def index():
   return render_template('index.html')
```



Things We Did Not See

Things we didn't see

Optional and variable number of parameters in functions (without default)

Generator functions: Standard ways to create (potentially infinite) iterable objects (the **yield** operator)

Decorators: Functions that can be attached to other functions and that can control their execution (e.g. add code before / after)

Assertions: Ways to check specific conditions in the code and raise exceptions if they are not verified.

...

To be seen in labs

Object Oriented Programming

Exercises on Exceptions

Fix Fest Debut 🥳