Computational Techniques (SSFM)

An introduction to Python

Prof. Stefano Carrazza

University of Milan and INFN Milan

Outline

Outline

Contacts:

- **Lecturers:** Stefano Carrazza, Stefano Ferrari (titolare)
- **E-mails:** stefano.carrazza@unimi.it, stefano.ferrari@unimi.it

Outline

- **Topics covered in this lecture:**
  - Python setup and language overview
  - Numpy basics
  - Matplotlib basics
  - Pydicom basics
- **References:** A Byte of Python `https://python.swaroopch.com/`
- **Examination:** send the final notebook with exercises to `stefano.carrazza@unimi.it` before the conclusion of the lectures.

Why Python?

Images

Definition:

An **image** is an artifact that depicts visual perception.

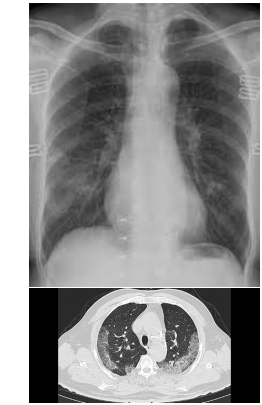
Devices used to capture images:

- *Optical devices:* cameras, mirrors, lenses, telescopes, microscopes
- *Natural objects and phenomena:* human eye

Examples in medicine

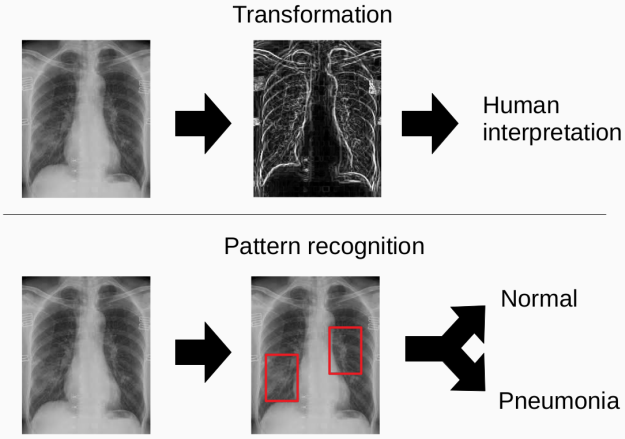
Types of medical imaging:

- Radiography
- Magnetic resonance imaging
- Tomography
- Nuclear Medicine
- Ultrasound
- Echocardiography
- Functional near-infrared spectroscopy
- Magnetic Particle Imaging
- Elastography
- Photoacoustic imaging



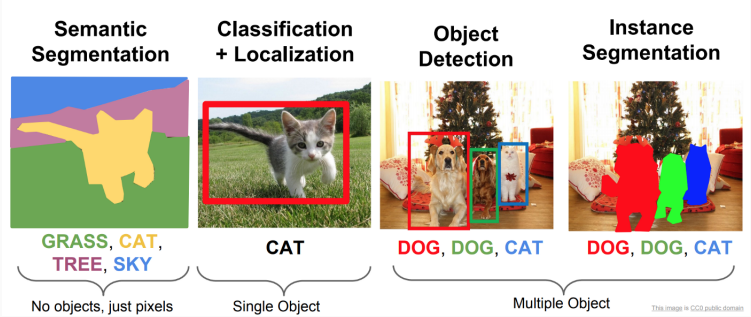
Digital images treatment

Image processing techniques:



Digital images treatment

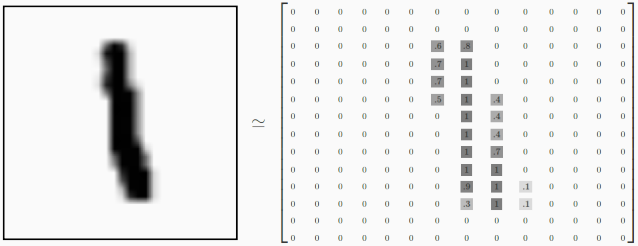
Pattern recognition algorithms:



Digital images treatment

Nowadays images are converted to digital images representation.

Example: consider a simple image from the MNIST dataset:

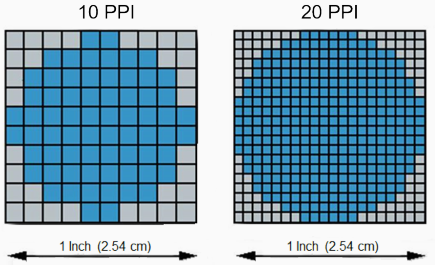


Digital images provide the possibility to:

- Store and transfer data images across computers.
- Apply and implement image processing algorithms.

Digital image representation

A digital image is an image composed of pixels, each with finite, discrete quantities of numeric representation for its intensity or gray level.



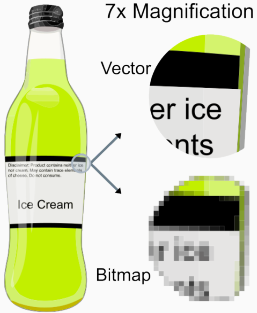
We can distinguish at least two properties:

- Image resolution: measure of pixels in the display (width x height).
- Image density: pixels per inch (PPI) or dot's per inch (DPI).

Raster vs vector images

Types of images:

- Raster/bitmapped image: resolution is **fixed**.
  Typical examples are png, jpeg, gif...
- Vector image: resolution is controlled by **vector coordinates**.
  Typical examples are pdf, svg, eps...



Digital image formats

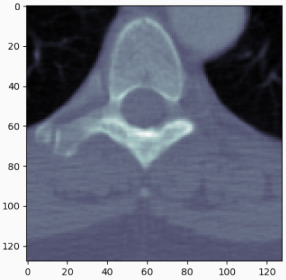
Digital Imaging and Communications in Medicine, **DICOM**, is the standard for medical imaging information and related data.

Pro:

- Standard format
- Includes structured data

Con:

- Require specific parser



Out:

```
Filename.........: /home/circleci/project/pydicom/data/test_files/CT_small.dcm
Storage type.....: 1.2.840.10008.5.1.4.1.1.2

Patient's name...: CompressedSamples, CT1
Patient id.......: 1CT1
Modality.........: CT
Study Date.......: 20040119
Image size.......: 128 x 128, 32768 bytes
Pixel spacing....: [0.661468, 0.661468]
Slice location...: -77.2040634155
```

Digital image formats

Computational Techniques $\Rightarrow$ **Programming language required.**

Introducing Python

Introducing Python

Python

Paradigm	Multi-paradigm: object-oriented,[1] procedural (imperative), functional, structured, reflective
Designed by	Guido van Rossum
Developer	Python Software Foundation
First appeared	20 February 1991; 30 years ago[2]
Stable release	3.10.0[3] ✓ / 4 October 2021; 56 days ago
Preview release	3.11.0a2[4] ✓ / 5 November 2021; 24 days ago
Typing discipline	Duck, dynamic, strong typing;[5] gradual (since 3.5, but ignored in CPython)[6]
OS	Windows, Linux/UNIX, macOS and more[7]
License	Python Software Foundation License
Filename extensions	.py, .pyi, .pyc, .pyd, .pyo (prior to 3.5),[8] .pyw, .pyz (since 3.5)[9]
Website	www.python.org
Major implementations	CPython, PyPy, Stackless Python, MicroPython, CircuitPython, IronPython, Jython
Dialects	Cython, RPython, Starlark[10]
Influenced by	ABC,[11] Ada,[12] ALGOL 68,[13] APL,[14] C,[15] C++,[16] CLU,[17] Dylan,[18] Haskell,[19] Icon,[20] Java,[21] Lisp,[22] Modula-3,[24] Perl, Standard ML[24]
Influenced	Apache Groovy, Boo, Cobra, CoffeeScript,[25] D, F#, Genie,[26] Go, JavaScript,[25][26] Julia,[27] Nim, Ring,[28] Ruby,[29] Swift[30]

Python Programming at Wikibooks

C++

Logo endorsed by Standard C++

Paradigms	Multi-paradigm: procedural, functional, object-oriented, generic, modular
Family	C
Designed by	Bjarne Stroustrup
Developer	ISO/IEC JTC1 (Joint Technical Committee 1) / SC22 (Subcommittee 22) / WG21 (Working Group 21)
First appeared	1985; 36 years ago
Stable release	C++20 (ISO/IEC 14882:2020) / 15 December 2020; 11 months ago
Preview release	C++23 / 23 October 2021; 38 days ago
Typing discipline	Static, nominative, partially inferred
OS	Cross-platform
Filename extensions	.C, .cc, .cpp, .cxx, .c++, .h, .H, .hh, .hpp, .hxx, .h++
Website	isocpp.org
Major implementations	GCC, LLVM Clang, Microsoft Visual C++, Embarcadero C++Builder, Intel C++ Compiler, IBM XL C++, EDG
Influenced by	Ada,[1] ALGOL 68, C, CLU,[1] ML, Mesa,[1] Modula-2,[1] Simula, Smalltalk[1]
Influenced	Ada 95, C#,[2] C99, Chapel,[3] Clojure,[4] D, Java,[5] JS++,[6] Lua, Nim,[7] Objective-C++, Perl, PHP, Python,[8] Rust, Seed7

C++ Programming at Wikibooks

Some famous Python libraries:

Most popular programming language

Python is the preferred programming language for ML applications, furthermore there are several modules which simplifies data analysis tasks.

Most popular public deep learning frameworks

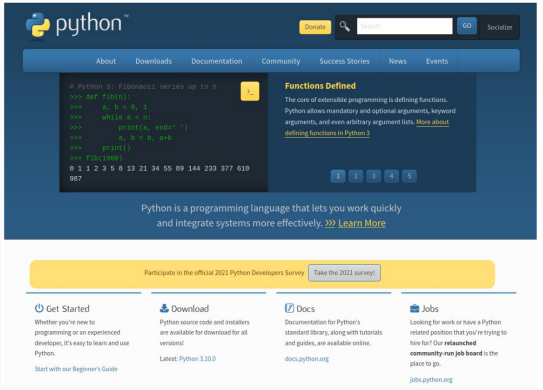
Some frameworks for deep learning deployment:

- Keras: a Python deep learning library.
- TensorFlow: library for numerical computation with data flow graphs.
- PyTorch: a DL framework for fast, flexible experimentation.
- Caffe: speed oriented deep learning framework.
- Theano: a Python library for optimization.
- MXNet: deep learning framework for neural networks.
- CNTK: Microsoft Cognitive Toolkit.
- scikit-learn: general machine learning package.

How to install python?

Python comes pre-installed in UNIX-based systems.

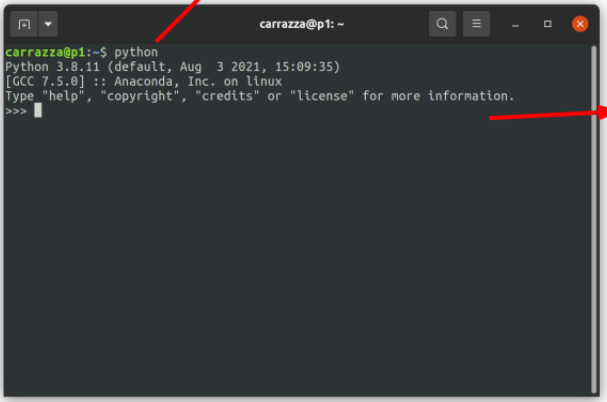
Anyway there are official binaries for all platforms in `https://www.python.org`.



Warning: python3 != python2. Use python3.

How to install python?

Launch python command.

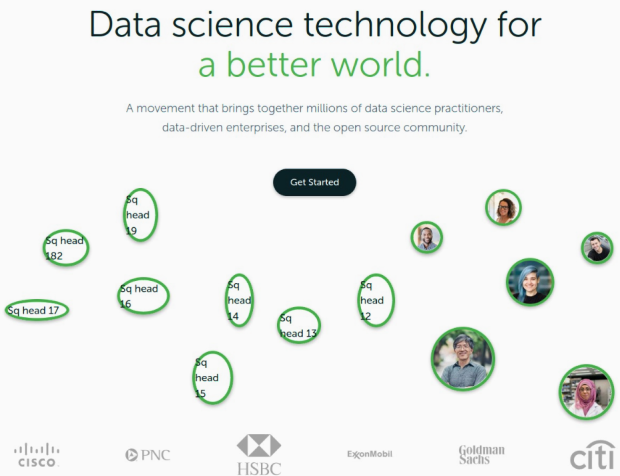


Interactive shell

Type: exit() or CRTL+D

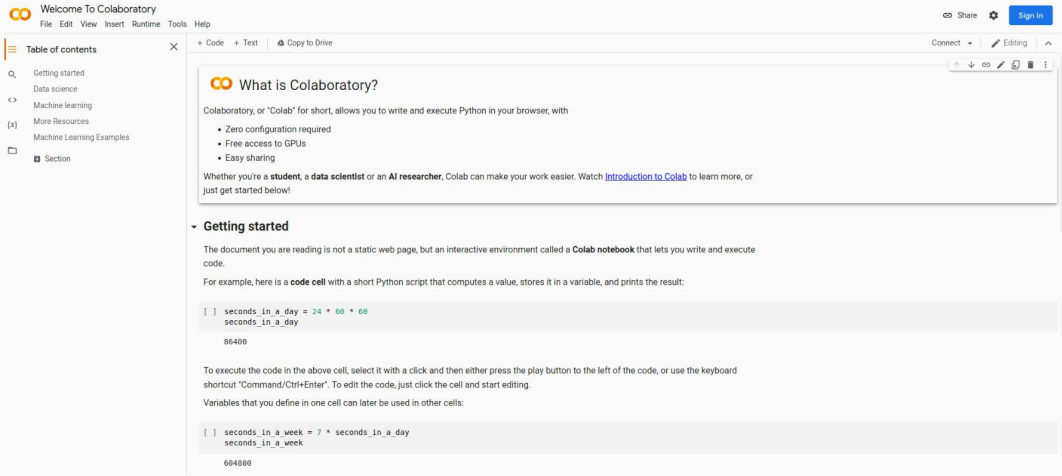
Anaconda

Anaconda and miniconda are other simple alternatives for a full integrated development environment: `https://anaconda.org`



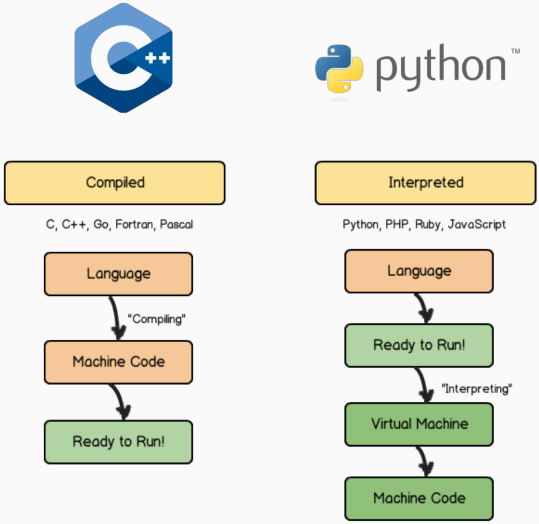
Jupyter notebooks

You could even code directly in your browser thanks to Jupyter Notebooks, and services like Google Colab: `https://colab.research.google.com`



The Python interpreter

The Python interpreter



The Python interpreter

Pro

- Fast prototyping
- Large community, huge number of libraries

Cons

- Bad performance for intensive computation using native python primitives.
- Installation process and dependency resolver not trivial.

The Python modules

Third-party libraries can be imported easily in your code:

Numerical Python
https://numpy.org/

Visualization with Pythor
https://matplotlib.org/

Example:

```
pip install numpy
pip install matplotlib
pip install pydicom
```

The Python language

The Python language

Printing to terminal:

```
print("Hello World!")
```

Define a script `myscript.py`:

```
#!/usr/bin/env python
print("Hello World!")
```

Or "`chmod +x script.py`" and then "`python myscript.py`".

The Python language

Variable allocation (dynamic typing):

```
a = 1          # integer
b = -16.5      # double
c = "Hello"    # string
d = True       # bool
e = 1+5j       # complex
```

Printing:

```
print(a, b, c, d, e)
print(f'{a} {b} {d} {e}') # f-string
```

Lists:

```
e = []        # empty list
f = [1, 2]    # initialized list

f.append(5)   # append to list
f += [6]      # append to list

list_length = len(f) # returns 4
```

Dictionaries:

```
g = {'key1': 5,
     'key2': [1, 2]}

# getter
v = g.get('key1')

# setter
g['key3'] = 6
```

The Python language

Conditionals:

```
if condition1:
  <code>
elif condition2:
  <code>
else:
  <code>
```

Example:

```
a = input("positive integer:")
test = a > 0
if test:
  print("input is positive")
else:
  print("input is negative")
```

The Python language

Control flow statement for:

```
n = 10
for i in range(n):
  <code>
  break
  continue
  <code>
```

Control flow statement while:

```
condition = True
while condition:
  <code>
  break
  continue
  <code>
```

The Python language

Functions:

```
def myfunction(arguments):
  r = <code>
  return r
```

Main-like script:

```
def myfunction(x, p):
  return x**p

if __name__ == "__main__":
  v = myfunction(5, 2)
  print(v)
```

The Python language

Classes:

```
class MyClass:
  def __init__(self, name): # constructor
      self.variable = ...

  def sample(self): # method
      return ...
```

Inheritance:

```
class ChildClass(MyClass):
  def __init__(self):
    super().__init__() # executes MyClass constructor
    self.variable *= 2
```

The Python language

```python
import numpy as np

def myfunction(x):
  """Evaluates exponential of x^2"""
  return np.exp(-np.square(x))

def main():
  """Main function demo."""
  a = 5        # int
  b = "hello" # string
  c = 6j       # complex
  results = []
  for ix in range(10):
    results.append(myfunction(ix))

  if results[0] != 1:
      print("Error")

if __name__ == "__main__":
  main()
```

Exercises 1-7: `https://github.com/scarrazza/ssfm_python_tutorial`

Numpy basics

Arrays

Let's suppose we have the matrix (32bit floats):

$$a = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ -1 & 0 & 3 \end{pmatrix}$$

We allocate this matrix with:

```
import numpy as np
a = np.array([[1, 0, 0], [0, 1, 2], [-1, 0, 3]], dtype=np.float32)

type(a) # <class 'numpy.ndarray'>
a.shape # returns (3,3)
a.ndim  # 2
a.dtype # dtype('float32')
a.flatten() # [1, 0, 0, 0, 1, 2, -1, 0, 3]
```

Numpy provides several dtypes: int32, int64, complex32, complex64, float32, float64...

Arrays

```
import numpy as np

# building array with zeros:
z = np.zeros(5, dtype=np.int32)
# array([0, 0, 0, 0, 0], dtype=int32)

# building array with ones
o = np.ones((5,1), dtype=np.float64)

# reshaping array
w = o.reshape(1, 5)
# array([[1., 1., 1., 1., 1.]])
```

Math operations

```
import numpy as np
a = np.array([1, 2, 3, 5])
b = np.ones(4)

c = a + b
# array([2., 3., 4., 6.])

d = c ** 2
e = 2 * np.sin(d)
# array([-1.51360499,  0.82423697, -0.57580663, -1.98355771])

e > 0
# array([False,  True, False, False])
```

Linear algebra

```
import numpy as np

A = np.eye(2) # [[1, 0], [0, 1]]
B = np.array([[2, 1], [3, 4]])

# elementwise product
A * B # [[2, 0], [0, 4]]

# matrix product
A @ B # or A.dot(B)
# [[2, 1], [3, 4]]

# transpose matrix
B.T
# [[2, 3],[1, 4]]
```

Slicing

```
import numpy as np

C = np.array([[2, 1], [3, 4]])

m = 0
C[m, m]  # 2
C[:, m]  # [2, 3]
C[m, :]  # [2, 1]
C[m, -1] # 1
C[-1, m] # 3

C.flatten()[0]   # 2
C.flatten()[-1]  # 4
C.flatten()[0:2] # [2, 1]
```

Copy

```
import numpy as np

a = np.array([[2, 1], [3, 4]])

# no copy
b = a # a and b are the same ndarray

# shallow copy
c = a.view()
c[0, 0] = -1 # a's data change
c = c.reshape(4) # a's shape doesn't change

# deep copy
d = a.copy()
```

Input/Output

```
import numpy as np

# store array
a = np.array([[2, 1], [3, 4]])
np.save("myarray.npy", a)     # binary format
np.savetxt("myarray.txt", a) # or text format

# read from file
b = np.load("myarray.npy")
b = np.loadtxt("myarray.txt")
```

Numpy

Exercises 8-9: `https://github.com/scarrazza/ssfm_python_tutorial`

Matplotlib basics

Matplotlib

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

x = np.linspace(0, 2, 100) # 100 points linearly spaced [0,2]

plt.plot(x, x**2, label="quadratic")

plt.title("Quadratic curve")
plt.xlabel("x label")
plt.ylabel("y label")
plt.legend()

plt.savefig("myplot.png")
plt.show() # keeps the plot open if script
```

Matplotlib

https://matplotlib.org/cheatsheets/_images/cheatsheets-1.png

Exercises 10-12: https://github.com/scarrazza/ssfm_python_tutorial

Pydicom basics

How to read a DICOM file

```
import pydicom

# load dicom
ds = pydicom.dcmread("CT_file.dcm")

# print information
ds.PatientName      # patient's name
ds.PatientID        # patient id
ds.StudyDate        # study date
ds.Rows, ds.Columns # image size

# get image
ds.pixel_array # return numpy array with pixel values

plt.imshow(ds.pixel_array, cmap=plt.cm.gray) # plot image
```

Pydicom

Exercises 13: `https://github.com/scarrazza/ssfm_python_tutorial`