Deep Learning - lab 1

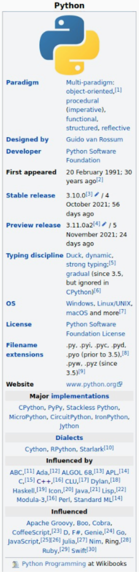
An introduction to Python

Prof. Stefano Carrazza

University of Milan and INFN Milan

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]

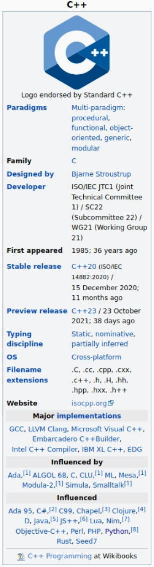
Influenced by

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,[16] Perl, Standard ML[14]

Influenced

Apache Groovy, Boo, Cobra, CoffeeScript,[23] D, F#, Genie,[24] 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

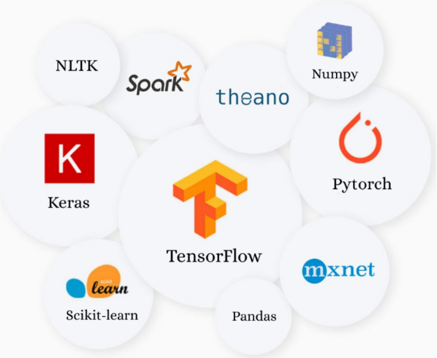
Ada,[1] ALGOL 68, C, CLU,[1] ML, Mesa,[1] Modula-2,[1] Simula, Smalltalk[1]

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

SciPy

NumPy

matplotlib

IP[y]: IPython
Interactive Computing

pandas $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

scikit learn

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

Most popular public DL frameworks

Some frameworks for deep learning deployment:

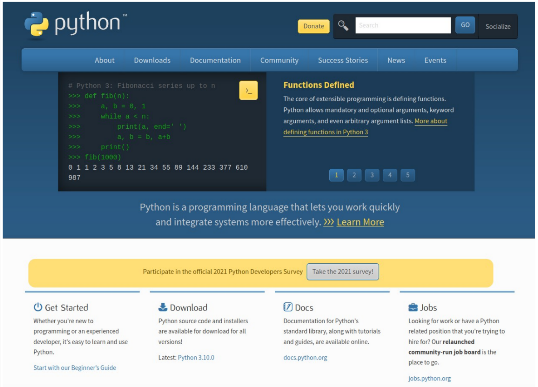
- Keras: a Python deep learning library.
- TensorFlow: library for numerical computation with data flow graphs.
- PyTorch: a DL framework for fast, flexible experimentation.
- Caffe: speed oriented deep learning framework.
- Theano: a Python library for optimization.
- MXNet: deep learning framework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

Why use public codes? → builtin models and automatic differentiation

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`.



```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
987
```

Functions Defined

The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. More about defining functions in Python 3

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Participate in the official 2021 Python Developers Survey

Get Started

Whether you're new to programming or an experienced developer, it's easy to learn and use Python.

Start with our Beginner's Guide

Download

Python source code and installers are available for download for all versions!

Latest: Python 3.10.0

Docs

Documentation for Python's standard library, along with tutorials and guides, are available online.

docs.python.org

Jobs

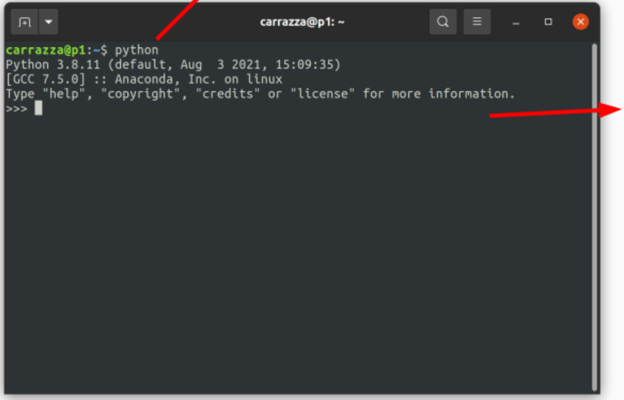
Looking for work or have a Python related position that you're trying to hire for? Our **relaunched community-run job board** is the place to go.

jobs.python.org

Warning: python3 != python2. Use python3.

How to install python?

Launch python command.



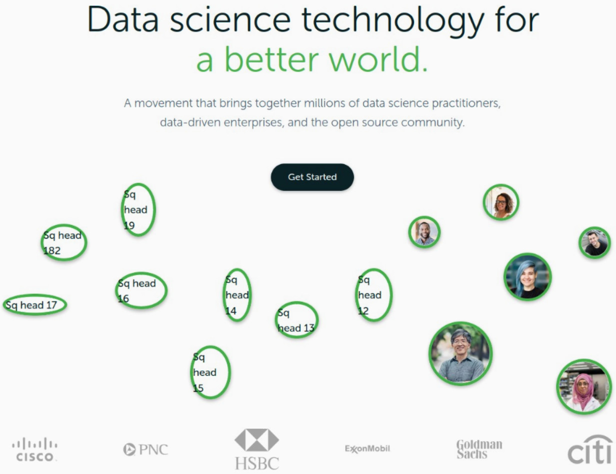
```
carrazza@p1:~$ python
Python 3.8.11 (default, Aug  3 2021, 15:09:35)
[GCC 7.5.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Interactive shell

Type: exit() or CRTL+D

Anaconda

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

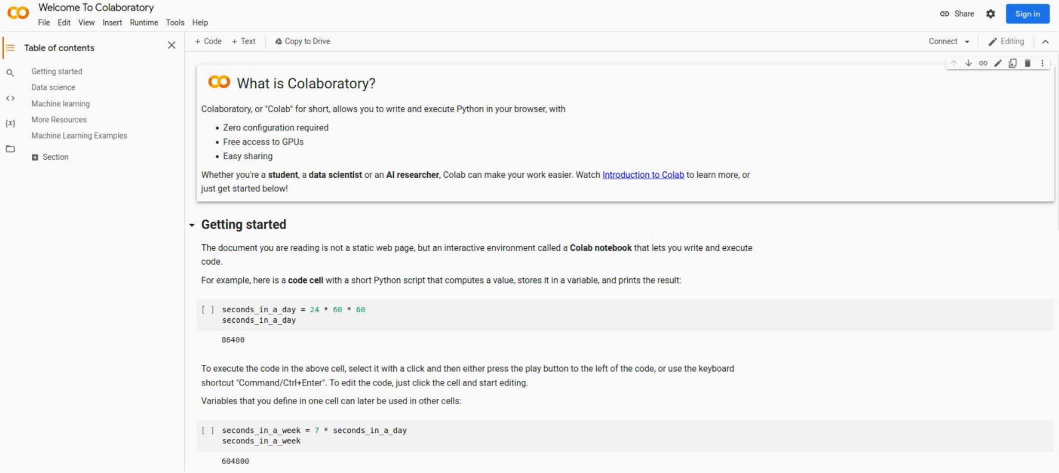


Data science technology for
a better world.

A movement that brings together millions of data science practitioners, data-driven enterprises, and the open source community.

Jupyter notebooks

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



What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!

Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
    seconds_in_a_day

    86400
```

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

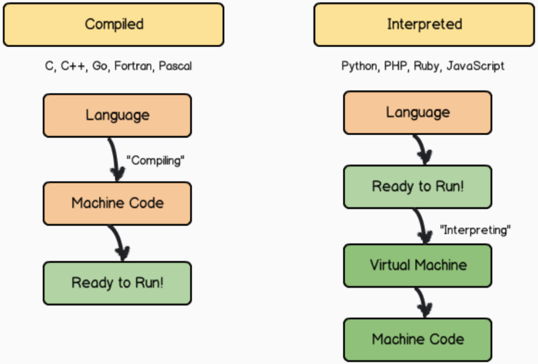
Variables that you define in one cell can later be used in other cells:

```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
    seconds_in_a_week

    604800
```

The Python interpreter

The Python interpreter



Compiled

C, C++, Go, Fortran, Pascal

Language

"Compiling"

Machine Code

Ready to Run!

Interpreted

Python, PHP, Ruby, JavaScript

Language

Ready to Run!

"Interpreting"

Virtual Machine

Machine Code

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:

NumPy

Numerical Python
https://numpy.org/

matplotlib

Visualization with Python
https://matplotlib.org/

Example:

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

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):

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

Printing:

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

The Python language

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
```

The Python language

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()
```