DIT gentle introduction to Python

v 2.3 (PhD) January–February 2024

Alberto Barrón-Cedeño

Alma Mater Studiorum-Università di Bologna
a.barron@unibo.it @_albarron_

29/01/2024

Table of Contents

Basics

What is a programming language?

A programming language is just a language. . .

What is a programming language?

A programming language is just a language. . .

> *A formal language comprising a set of instructions that produce various kinds of output [given an input]*

https://en.wikipedia.org/wiki/Programming_language
(from an old version of the article; I don't like the current definition)

What is a programming language?

A programming language is just a language. . .

> *A formal language comprising a set of instructions that produce various kinds of output [given an input]*

`https://en.wikipedia.org/wiki/Programming_language`
(from an old version of the article; I don't like the current definition)

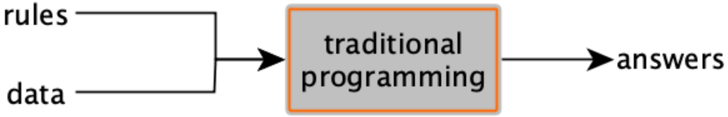


Diagram borrowed from L. Moroney's Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning

What is a programming language?

*Programming languages are used in computer programming to implement an algorithm**

https://en.wikipedia.org/wiki/Programming_language

What is a programming language?

*Programming languages are used in computer programming to implement an algorithm**

https://en.wikipedia.org/wiki/Programming_language

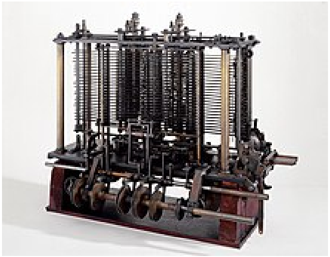
1983 USSR stamp commemorating al-Khwārizmī's (approximate) 1200th birthday

* derived from the 9th century Persian Mathematician Muhammad ibn Mūsā al-Khwārizmī

The *first* programmer

A. Lovelace by 1840

Ada Lovelace[a] (Mathematician) published the first algorithm for Charles Babbage's analytical engine



[a]Lord Byron's daughter

Algorithms

Algorithm

A finite sequence of well-defined computer-implementable instructions, typically to solve a class of problems or to perform a computation

https://en.wikipedia.org/wiki/Algorithm

Algorithm Example: Find out if a number is odd or even*

*Adapted from https://www.c-programming-simple-steps.com/algorithm-examples.html

Algorithm Example: Find out if a number is odd or even*

Definitions

- A number is even if it can be divided by 2 without remainder
- A number is odd if it leaves a remainder when divided by 2

*Adapted from `https://www.c-programming-simple-steps.com/algorithm-examples.html`

Algorithm Example: Find out if a number is odd or even*

Definitions

- A number is even if it can be divided by 2 without remainder
- A number is odd if it leaves a remainder when divided by 2

Examples

Even numbers: 2, 4, 6, 8, etc.

Odd numbers: 1, 3, 5, 7, etc.

*Adapted from
https://www.c-programming-simple-steps.com/algorithm-examples.html

Algorithm Example: Find out if a number is odd or even*

Definitions

- A number is even if it can be divided by 2 without remainder
- A number is odd if it leaves a remainder when divided by 2

Examples

Even numbers: 2, 4, 6, 8, etc.

Odd numbers: 1, 3, 5, 7, etc.

Silly (useless) solution:

- Produce all possible even numbers and store them in *box* EVEN. Produce all possible odd numbers and store them *box* ODD.
- Given an input number, look for it in both boxes return the label of the one in which you found it

*Adapted from `https://www.c-programming-simple-steps.com/algorithm-examples.html`

Algorithm Example: Find out if a number is odd or even

Problem Definition

Input/Output

$\rightarrow$ an integer (data)

$\leftarrow$ even or odd (more data)

Diagram borrowed from
https://www.c-programming-simple-steps.com/algorithm-examples.html

Algorithm Example: Find out if a number is odd or even

Problem Definition

Input/Output

→ an integer (data)

← even or odd (more data)

Process

A series of instructions and routines

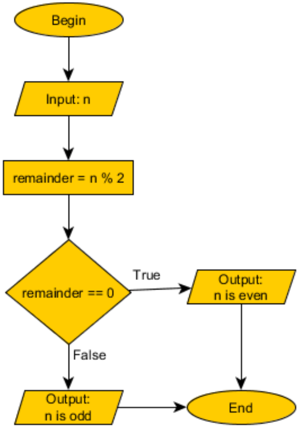
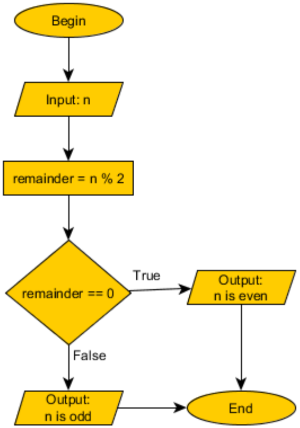


Diagram borrowed from
https://www.c-programming-simple-steps.com/algorithm-examples.html

Algorithm Example: Find out if a number is odd or even

From the algorithm to the implementation



```
if n%2 == 0:
  print('even')
else:
  print('odd')
```

Programming languages

History of (some) flagship languages (1/2)

year	**language**	**highlights**
1957	Fortran	Compiled, imperative
1959	Lisp*	Object-oriented, popular in AI, recursive functions
1964	Basic*	Procedural, object-oriented (“goto”)
1970	Pascal*	Imperative, procedural, lists, trees
1972	C*	Procedural, recursion, static type system
1983	C^{++}*	Object-oriented, compiled, functional

* language I “speak” (or “spoke” at some point in time)

History of (some) flagship languages (2/2)

year	language	highlights
1989	Python*	Interpreted, object-oriented, code readability
1995	Java*	Compiled, object-oriented
1995	Javascript	Just-in-time-compiled, object-oriented, WWW
1995	PHP*	Scripting, Web-oriented
2001	V. Basic.NET	Object-oriented, .NET framework
2009	Go	Compiled, C-like (safer)

* language I "speak" (or "spoke" at some point in time)

Python

(Among other things), python is. . .

General-purpose
Applicable across application domains

Python

(Among other things), python is. . .

General-purpose
Applicable across application domains

High-level
Strong abstraction from the computer (hardware)

Python

(Among other things), python is. . .

General-purpose
Applicable across application domains

High-level
Strong abstraction from the computer (hardware)

Interpreted
No previous compilation into machine-level instructions necessary

Python

(Among other things), python is. . .

General-purpose
Applicable across application domains

High-level
Strong abstraction from the computer (hardware)

Interpreted
No previous compilation into machine-level instructions necessary

(Not-necessarily) object-oriented
An object contains data (attributes) and procedures (methods)

Python

Some notable features

- Elegant syntax (indentation-based) $\rightarrow$ easy to read

https://wiki.python.org/moin/BeginnersGuide/Overview

Python

Some notable features

- Elegant syntax (indentation-based) $\rightarrow$ easy to read
- Simple and ideal for prototyping

`https://wiki.python.org/moin/BeginnersGuide/Overview`

Python

Some notable features

- Elegant syntax (indentation-based) $\rightarrow$ easy to read
- Simple and ideal for prototyping
- It has a large standard library for diverse tasks (e.g., web servers, text search and processing, file reading/modifying)

https://wiki.python.org/moin/BeginnersGuide/Overview

Python

Some notable features

- Elegant syntax (indentation-based) → easy to read
- Simple and ideal for prototyping
- It has a large standard library for diverse tasks (e.g., web servers, text search and processing, file reading/modifying)
- Interactive mode → continuous snippet testing

`https://wiki.python.org/moin/BeginnersGuide/Overview`

Python

Some notable features

- Elegant syntax (indentation-based) → easy to read
- Simple and ideal for prototyping
- It has a large standard library for diverse tasks (e.g., web servers, text search and processing, file reading/modifying)
- Interactive mode → continuous snippet testing
- Extendable with modules in compiled languages (e.g., C++)

https://wiki.python.org/moin/BeginnersGuide/Overview

Python
Some notable features

- Elegant syntax (indentation-based) $\rightarrow$ easy to read
- Simple and ideal for prototyping
- It has a large standard library for diverse tasks (e.g., web servers, text search and processing, file reading/modifying)
- Interactive mode $\rightarrow$ continuous snippet testing
- Extendable with modules in compiled languages (e.g., C++)
- Multi-platform (e.g., Mac OS X, GNU Linux, Unix, MS Windows)

https://wiki.python.org/moin/BeginnersGuide/Overview

Python

Some notable features

- Elegant syntax (indentation-based) $\rightarrow$ easy to read
- Simple and ideal for prototyping
- It has a large standard library for diverse tasks (e.g., web servers, text search and processing, file reading/modifying)
- Interactive mode $\rightarrow$ continuous snippet testing
- Extendable with modules in compiled languages (e.g., C++)
- Multi-platform (e.g., Mac OS X, GNU Linux, Unix, MS Windows)
- Free: zero-cost to download/use; open-source license

https://wiki.python.org/moin/BeginnersGuide/Overview

Python

Some notable features

- Elegant syntax (indentation-based) → easy to read
- Simple and ideal for prototyping
- It has a large standard library for diverse tasks (e.g., web servers, text search and processing, file reading/modifying)
- Interactive mode → continuous snippet testing
- Extendable with modules in compiled languages (e.g., C++)
- Multi-platform (e.g., Mac OS X, GNU Linux, Unix, MS Windows)
- Free: zero-cost to download/use; open-source license
- Large and friendly community

https://wiki.python.org/moin/BeginnersGuide/Overview

Python

Some notable features

- Elegant syntax (indentation-based) → easy to read
- Simple and ideal for prototyping
- It has a large standard library for diverse tasks (e.g., web servers, text search and processing, file reading/modifying)
- Interactive mode → continuous snippet testing
- Extendable with modules in compiled languages (e.g., C++)
- Multi-platform (e.g., Mac OS X, GNU Linux, Unix, MS Windows)
- Free: zero-cost to download/use; open-source license
- Large and friendly community
- **Top alternative for deep learning**

https://wiki.python.org/moin/BeginnersGuide/Overview

Python

Some programming-language features

- A variety of basic data types are available:[1]
  - numbers (floating point, complex, integers)
  - strings (both ASCII and Unicode)
  - Lists
  - Dictionaries

[1] Later today

Python

Some programming-language features

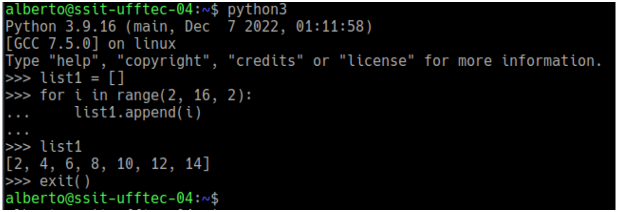
- A variety of basic data types are available:[1]
  - numbers (floating point, complex, integers)
  - strings (both ASCII and Unicode)
  - Lists
  - Dictionaries
- It supports object-oriented programming
- Code can be grouped into modules and packages

[1]Later today

Python

Some ways to code/launch a python program

[UNIX , GNU Linux , MacOS , Windows] terminal

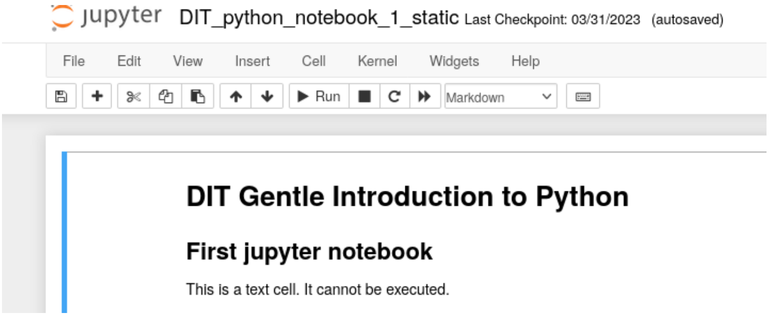


```
alberto@ssit-ufftec-04:~$ python3
Python 3.9.16 (main, Dec  7 2022, 01:11:58)
[GCC 7.5.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> list1 = []
>>> for i in range(2, 16, 2):
...     list1.append(i)
...
>>> list1
[2, 4, 6, 8, 10, 12, 14]
>>> exit()
alberto@ssit-ufftec-04:~$
```

Python

Some ways to code/launch a python program

Web browser: local, online, on Google's colab


jupyter
DIT_python_notebook_1_static Last Checkpoint: 03/31/2023 (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Markdown

DIT Gentle Introduction to Python

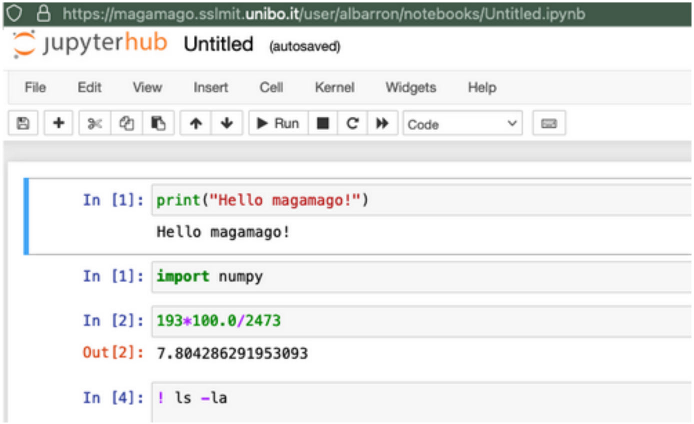
First jupyter notebook

This is a text cell. It cannot be executed.

Python

Some ways to code/launch a python program

From your web browser on DIT's magamago (remotely online)[2]


https://magamago.sslmit.unibo.it/user/albarron/notebooks/Untitled.ipynb
jupyterhub Untitled (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

```
In [1]: print("Hello magamago!")
        Hello magamago!

In [1]: import numpy

In [2]: 193*100.0/2473
Out[2]: 7.804286291953093

In [4]: ! ls -la
```

[2]Open to advanced students only

Enough! Let us look at some code!

Baby steps into coding

Google's colab

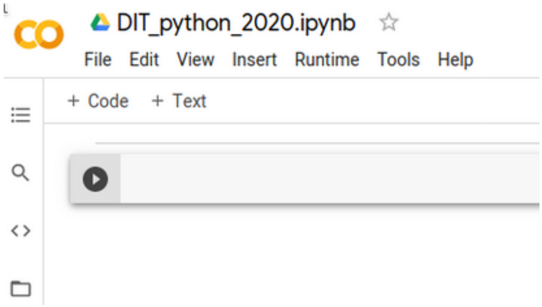
a free Jupyter notebook environment that runs in the cloud and stores its notebooks on Google Drive

`https://colab.research.google.com`

Google's colab

a free Jupyter notebook environment that runs in the cloud and stores its notebooks on Google Drive

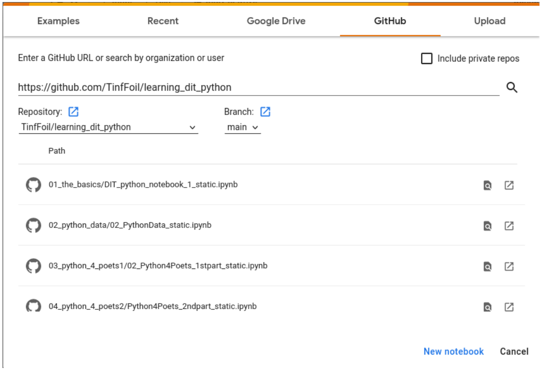
`https://colab.research.google.com`



Our first jupyter notebook

Google's colab: baby steps

1. Visit `https://colab.research.google.com`
2. Click on Github
3. Type (or paste) `https://github.com/TinfFoil/learning_dit_python`
4. Press search
5. Select **DIT_python_notebook_1_static.ipynb**



Enter a GitHub URL or search by organization or user

☐ Include private repos

https://github.com/TinfFoil/learning_dit_python

Repository: TinfFoil/learning_dit_python

Branch: main

Path

01_the_basics/DIT_python_notebook_1_static.ipynb

02_python_data/02_PythonData_static.ipynb

03_python_4_poets1/02_Python4Poets_1stpart_static.ipynb

04_python_4_poets2/Python4Poets_2ndpart_static.ipynb

Baby Steps

What we know so far

input/output

- print() displays stuff to the screen
- input() captures information from the user

Baby Steps

What we know so far

input/output

- print() displays stuff to the screen
- input() captures information from the user

variables

x = 5	x is a variable we assign values to a variable with = (aka store information)

Baby Steps

What we know so far

input/output

- print() displays stuff to the screen
- input() captures information from the user

variables

x = 5	x is a variable we assign values to a variable with = (aka store information)
x = 5	x is an integer
x = 5.5	x is a float
x = 'ciao'	x is a string
x = "ciao"	x is also a string
x = '5'	x is what?

Baby Steps

What we know so far

input/output

- print() displays stuff to the screen
- input() captures information from the user

variables

x = 5	x is a variable we assign values to a variable with = (aka store information)
x = 5	x is an integer
x = 5.5	x is a float
x = ‘ciao’	x is a string
x = “ciao”	x is also a string
x = ‘5’	x is what?
x = x * 3	we can apply operators to variables and (re-)assign the output to a variable

Baby Steps

What we know so far

Basic formatting

```
# my code
x = 0
while x < 50:
  for i in range(x):
    print('x', end="")
  print()
  x += 1
```

- Comments start with #
- A line break is enough to close an instruction (in Java or C, we need ;)
- A colon opens a code snippet
- Indentation is crucial

Baby Steps

What we know so far

flow control – conditionals

```
if (condition):
    execute something
elif (condition):
    execute something
else:
    execute something
```

Only one of these three snippets is executed

Baby Steps

What we know so far

flow control – conditionals

```
if (condition):
  execute something
elif (condition):
  execute something
else:
  execute something
```

Only one of these three snippets is executed

```
if (condition):
  execute something
if (condition):
  execute something
else:
  execute something
```

How is this different?

Baby Steps

What we know so far

flow control – conditionals

```
if (condition):
  execute something
elif (condition):
  execute something
else:
  execute something
```

Only one of these three snippets is executed

```
if (condition):
  execute something
if (condition):
  execute something
else:
  execute something
```

How is this different?

flow control – loops

The code snippet will be executed during a number of iterations

Danger: a loop could run forever if there is a *bug*

```
for (iterator):
  execute something
```

```
while (condition):
  execute something
```

You know a lot already!

It is your turn to play with the notebook