

AI & Robotics

Introduction to Python: part two

Goals



The **junior-colleague**

- can set up a Python environment using pip, ... and conda.
- can set up a Jupyter environment.
- can log data to the console and a file.
- can describe the pros and cons to logging to the console and a file.
- can track changes in a file using `tail -f`.
- can create unit tests for Python.

Types of assignments



- [M] Mandatory
(To master the course materials.)
- [R] Recommended
(Advised in order to reach the goals.)
- [C] Challenges
(Ideal to get a deeper understanding.)
- [F] Fun
(Fun extra assignments.)

Python environments

Subscribe!

i

Info



Google Cloud
AI Adventures

Presents:

Which Python package manager
should you use?

Yufeng Guo
@YufengG



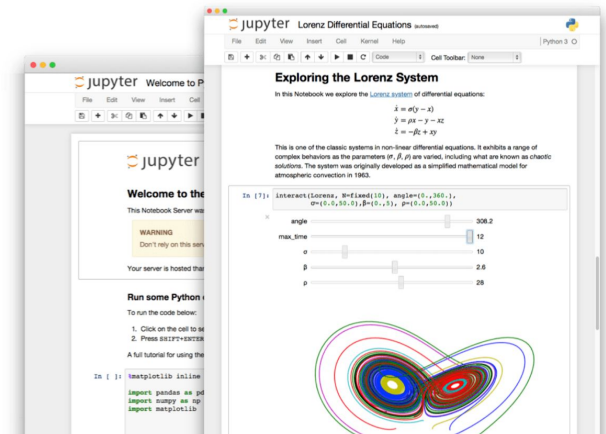
[SOURCE]

<https://www.youtube.com/watch?v=3J02sec99RM>

Python environments: Jupyter



[Install](#) [About Us](#) [Community](#) [Documentation](#) [NBViewer](#) [JupyterHub](#) [Widgets](#) [Blog](#)



The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

[Try it in your browser](#)

[Install the Notebook](#)



Language of choice

The Notebook has support for over 40 programming languages, including Python, R, Julia, and Scala.



Share notebooks

Notebooks can be shared with others using email, Dropbox, GitHub and the [Jupyter Notebook Viewer](#).



Interactive output

Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.



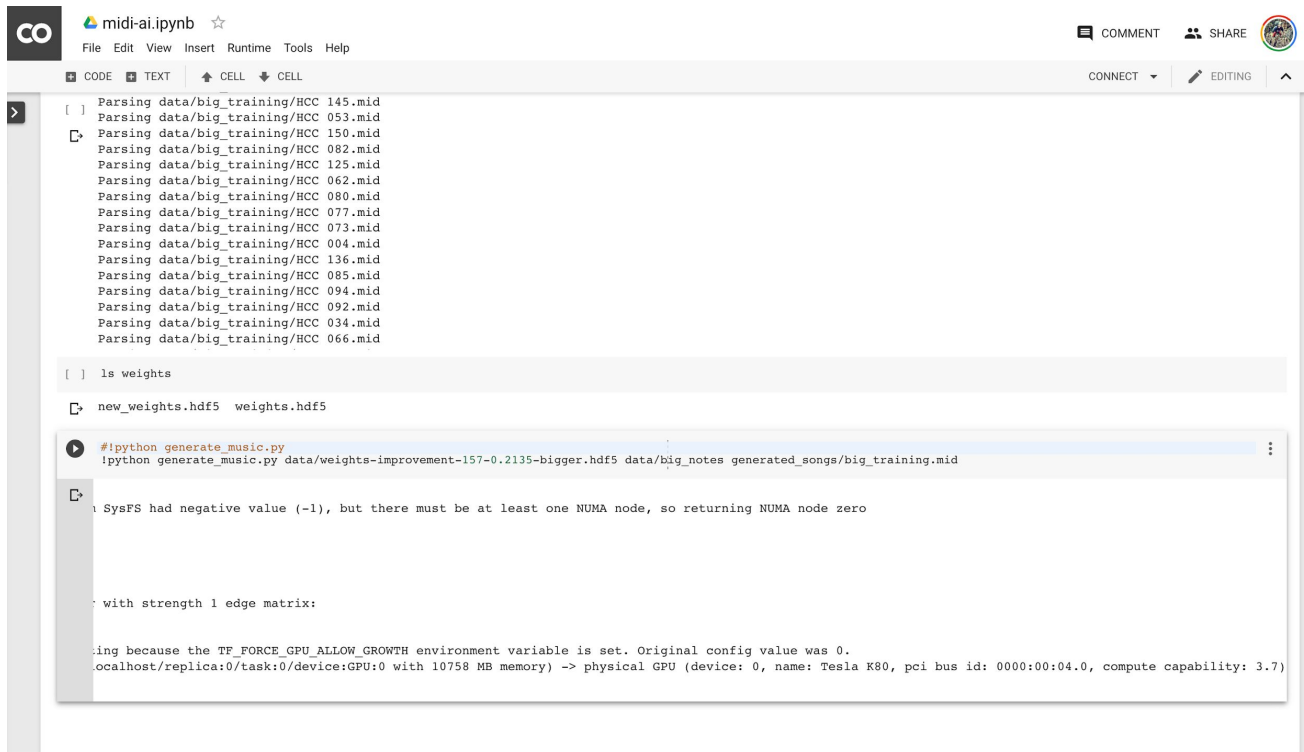
Big data integration

Leverage big data tools, such as Apache Spark, from Python, R and Scala. Explore that same data with pandas, scikit-learn, ggplot2, TensorFlow.

[[SOURCE](#)]

<https://jupyter.org>

Python environments: Colab



The screenshot shows a Google Colab notebook titled "midi-ai.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu is a toolbar with tabs for "CODE", "TEXT", "CELL", and "CELL". The notebook content is displayed in a code editor with a light gray background. The first cell contains a list of MIDI files being parsed from the "data/big_training/HCC" directory. The second cell shows the output of a command to list weights. The third cell shows the execution of a Python script to generate music, which results in a warning about SysFS and a message about the TF_FORCE_GPU_ALLOW_GROWTH environment variable.

```
[ ] Parsing data/big_training/HCC 145.mid
Parsing data/big_training/HCC 053.mid
[ ] Parsing data/big_training/HCC 150.mid
Parsing data/big_training/HCC 082.mid
Parsing data/big_training/HCC 125.mid
Parsing data/big_training/HCC 062.mid
Parsing data/big_training/HCC 080.mid
Parsing data/big_training/HCC 077.mid
Parsing data/big_training/HCC 073.mid
Parsing data/big_training/HCC 004.mid
Parsing data/big_training/HCC 136.mid
Parsing data/big_training/HCC 085.mid
Parsing data/big_training/HCC 094.mid
Parsing data/big_training/HCC 092.mid
Parsing data/big_training/HCC 034.mid
Parsing data/big_training/HCC 066.mid

[ ] ls weights
new_weights.hdf5 weights.hdf5

[ ] !python generate_music.py
!python generate_music.py data/weights-improvement-157-0.2135-bigger.hdf5 data/big_notes generated_songs/big_training.mid

SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

with strength 1 edge matrix:

ing because the TF_FORCE_GPU_ALLOW_GROWTH environment variable is set. Original config value was 0.
localhost/replica:0/task:0/device:GPU:0 with 10758 MB memory) -> physical GPU (device: 0, name: Tesla K80, pci bus id: 0000:00:04.0, compute capability: 3.7)
```

[SOURCE]

<https://colab.research.google.com>

Logging

Python » English » 3.7.2 » Documentation » Python HOWTOs »

Quick search [Go](#) | [previous](#) | [next](#) | [modules](#) | [index](#)

Table of Contents

- Logging Cookbook
 - Using logging in multiple modules
 - Logging from multiple threads
 - Multiple handlers and formatters
 - Logging to multiple destinations
 - Configuration server example
 - Dealing with handlers that block
 - Sending and receiving logging events across a network
 - Adding contextual information to your logging output
 - Using LoggerAdapters to impart contextual information
 - Using objects other than dicts to pass contextual information
 - Using Filters to impart contextual information
 - Logging to a single file from multiple processes
 - Using file rotation
 - Use of alternative formatting styles
 - Customizing

Logging Cookbook

Author: Vinay Sajip <vinay_sajip at red-dove dot com>

This page contains a number of recipes related to logging, which have been found useful in the past.

Using logging in multiple modules

Multiple calls to `logging.getLogger('someLogger')` return a reference to the same logger object. This is true not only within the same module, but also across modules as long as it is in the same Python interpreter process. It is true for references to the same object; additionally, application code can define and configure a parent logger in one module and create (but not configure) a child logger in a separate module, and all logger calls to the child will pass up to the parent. Here is a main module:

```
import logging
import auxiliary_module

# create logger with 'spam_application'
logger = logging.getLogger('spam_application')
logger.setLevel(logging.DEBUG)
# create file handler which logs even debug messages
fh = logging.FileHandler('spam.log')
fh.setLevel(logging.DEBUG)
# create console handler with a higher log level
ch = logging.StreamHandler()
ch.setLevel(logging.ERROR)
# create formatter and add it to the handlers
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
fh.setFormatter(formatter)
ch.setFormatter(formatter)
# add the handlers to the logger
logger.addHandler(fh)
logger.addHandler(ch)

logger.info('creating an instance of auxiliary module.Auxiliaryv')
```

[SOURCE]

<https://docs.python.org/3/howto/logging-cookbook.html>

Logging: PRO-TIPs

1. Events that are tracked can be handled in different ways. The simplest way of handling tracked events is to print them to the console. **Another common way is to write them to a disk file.**

→ [M] Google its benefits ;-)

2. `tail -f filename`

[SOURCE]

<https://docs.python.org/3/howto/logging-cookbook.html>

Getting started with testing in Python

[Start Here](#)[Tutorials](#)[Products](#)[More](#)



Real Python

Getting Started With Testing in Python

by Anthony Shaw · Oct 22, 2018 · 17 Comments · [best-practices](#) [intermediate](#) [testing](#)

Table of Contents

- [Testing Your Code](#)
 - [Automated vs. Manual Testing](#)
 - [Unit Tests vs. Integration Tests](#)
 - [Choosing a Test Runner](#)
- [Writing Your First Test](#)
 - [When to Write the Test](#)

— FREE Email Series —

Python Tricks

```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

Get Python Tricks »

 No spam. Unsubscribe any time.

All Tutorial Topics

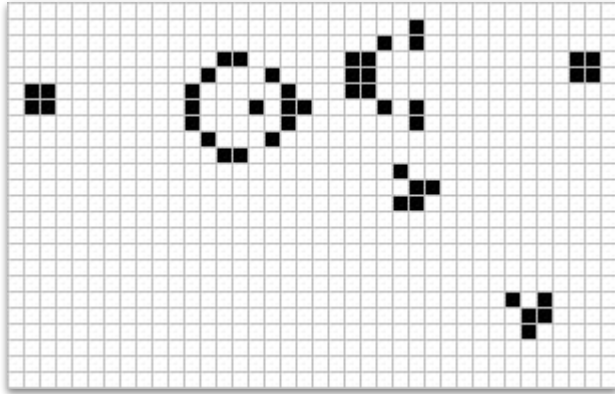
[advanced](#) [api](#) [basics](#) [best-practices](#) [community](#) [databases](#) [data-science](#) [devops](#) [django](#) [docker](#) [flask](#) [front-end](#) [intermediate](#) [machine-learning](#) [python](#) [testing](#) [tools](#) [web-dev](#) [web-scraping](#)


Improve Your Python 

[SOURCE]

<https://realpython.com/python-testing>

[R] Conway's game of life





WIKIPEDIA
The Free Encyclopedia

- Main page
- Contents
- Featured content
- Current events
- Random article
- Donate to Wikipedia
- Wikipedia store

Interaction

- Help
- About Wikipedia
- Community portal
- Recent changes
- Contact page

Tools

- What links here
- Related changes
- Upload file
- Special pages
- Permanent link
- Page information
- Wikidata item
- Cite this page

Print/export

- Create a book
- Download as PDF
- Printable version

In other projects

- Wikimedia Commons

Languages

- العربية
- Deutsch
- Español
- Français

Article [Talk](#)

Conway's Game of Life

From Wikipedia, the free encyclopedia

"Game of Life" redirects here. For other uses, see [Game of Life \(disambiguation\)](#).
"Conway game" redirects here. For Conway's surreal number game theory, see [Surreal number](#).

The **Game of Life**, also known simply as **Life**, is a cellular automaton devised by the British mathematician John Horton Conway in 1970.^[1]

The game is a zero-player game, meaning that its evolution is determined by its initial state, requiring no further input. One interacts with the Game of Life by creating an initial configuration and observing how it evolves, or, for advanced players, by creating patterns with particular properties.

Contents [hide]


- Rules
- Origins
- Examples of patterns
- Undecidability
- Self-replication
- Iteration
- Algorithms
- Variations
- Music
- Notable programs
- See also
- References
- External links

Rules [edit]

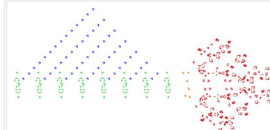
The universe of the *Game of Life* is an infinite, two-dimensional **orthogonal** grid of square *cells*, each of which is in one of two possible states, **alive** or **dead**, (or *populated* and *unpopulated*, respectively). Every cell interacts with its eight *neighbours*, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

- Any live cell with fewer than two live neighbors dies, as if by underpopulation.
- Any live cell with two or three live neighbors lives on to the next generation.
- Any live cell with more than three live neighbors dies, as if by overpopulation.
- Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

The initial pattern constitutes the *seed* of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed; births and deaths occur simultaneously, and the discrete moment



A single Gosper's glider gun creating "gliders"



A screenshot of a puffer-type breeder (red) that leaves glider guns (green) in its wake, which in turn create gliders (blue). (animation)

[SOURCE]

https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

[R] Assignments: Hangman



[R → C] Assignments: advent of code

```
Advent of Code [About] [Events] [Shop] [Log In]
$year=2018; [Calendar] [AoC++] [Sponsors] [Leaderboard] [Stats]

25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1

Our sponsors help
make Advent of
Code possible:

Clubhouse - The
first project
management
platform for
software
development that
brings everyone
on every team
together to build
better products.
AoC participants
get two free
months by signing
up at
r.clbh.se/mze9q9P
Also, we're
hiring!
```

[SOURCE]

<https://adventofcode.com>

[R] Assignments: Connect 4



[R] Assignments: Stratego



[R] Assignments: Battleship



[F] Assignments: scraper

```
39 #!/usr/env python3
38
37 import sys
36 import requests
35 from bs4 import BeautifulSoup
34 import os
33 import time
32 import re
31
30 def get_midi_file_urls(search_term):
29     urls = []
28     page = 1
27     next_page = True
26     while next_page:
25         next_page = False
24         url_begin = "http://www.midiworld.com/search/"
23         url_end = "/?q=" + search_term.replace(" ", "%20")
22
21         url = url_begin + str(page) + url_end
20         next_url = url_begin + str(page + 1) + url_end
19         response = requests.get(url)
18         html = response.text
17         soup = BeautifulSoup(html, "html.parser")
16
15         for li in soup.find_all("li"):
14             if "download" in str(li):
13                 for a in li.find_all("a", target="_blank"):
12                     urls.append(a.get("href"))
11
10         for link in soup.find_all('a'):
9             if link.get("href") == next_url:
8                 next_page = True
7                 page = page + 1
6                 break
5
4         time.sleep(0.01)
3
2         return urls
1
40 if __name__ == "__main__":
1     if len(sys.argv) != 2:
2         print("Usage: " + sys.argv[0] + " search_term")
3         print(" This will query midiworld.com and if there are any results")
4         print(" create a directory called \"search_term\" and store all the results in it.")
5         exit(-1)
6
7         search_term = sys.argv[1]
8
9         print("Searching...")
10        urls = get_midi_file_urls(search_term)
11
12        if not 0 < len(urls):
13            print("No files where found!")
14            exit(-1)
15
16        if 0 < len(urls) and not os.path.exists(search_term):
17            os.makedirs(search_term)
18
19        print("Downloading...")
20        for url in urls:
21            tries = 0
22            completed = False
23            while not completed:
24                try:
25                    response = requests.get(url, allow_redirects=True)
26                    disposition = response.headers['content-disposition']
27                    filename = re.findall("filename=(.*)" , disposition)[0] # Only the first occur
28                    print(" Saving " + filename + " into directory " + search_term)
29                    open(os.path.join(search_term, filename), 'wb').write(response.content)
30                    time.sleep(0.01)
31                    completed = True
32                except:
33                    tries = tries + 1
34                    print("Failed to download: " + filename + " (" + str(tries) + " tries)")
35                    time.sleep(0.1)
36                    if tries == 5:
37                        completed = True
38
39        print("Done")
40
```

[INFO]

Example <http://www.midiworld.com> scraper