AI & Robotics

Introduction to Python: part two

PXL AI & ROBOTICS LAB

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.)

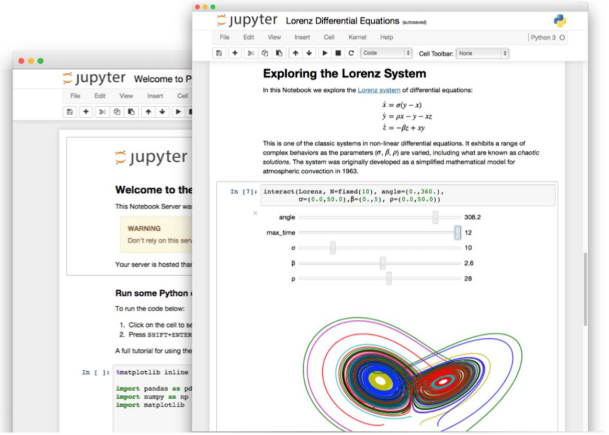
Python environments

[**SOURCE**]
https://www.youtube.com/watch?v=3J02sec99RM

Python environments: Jupyter

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

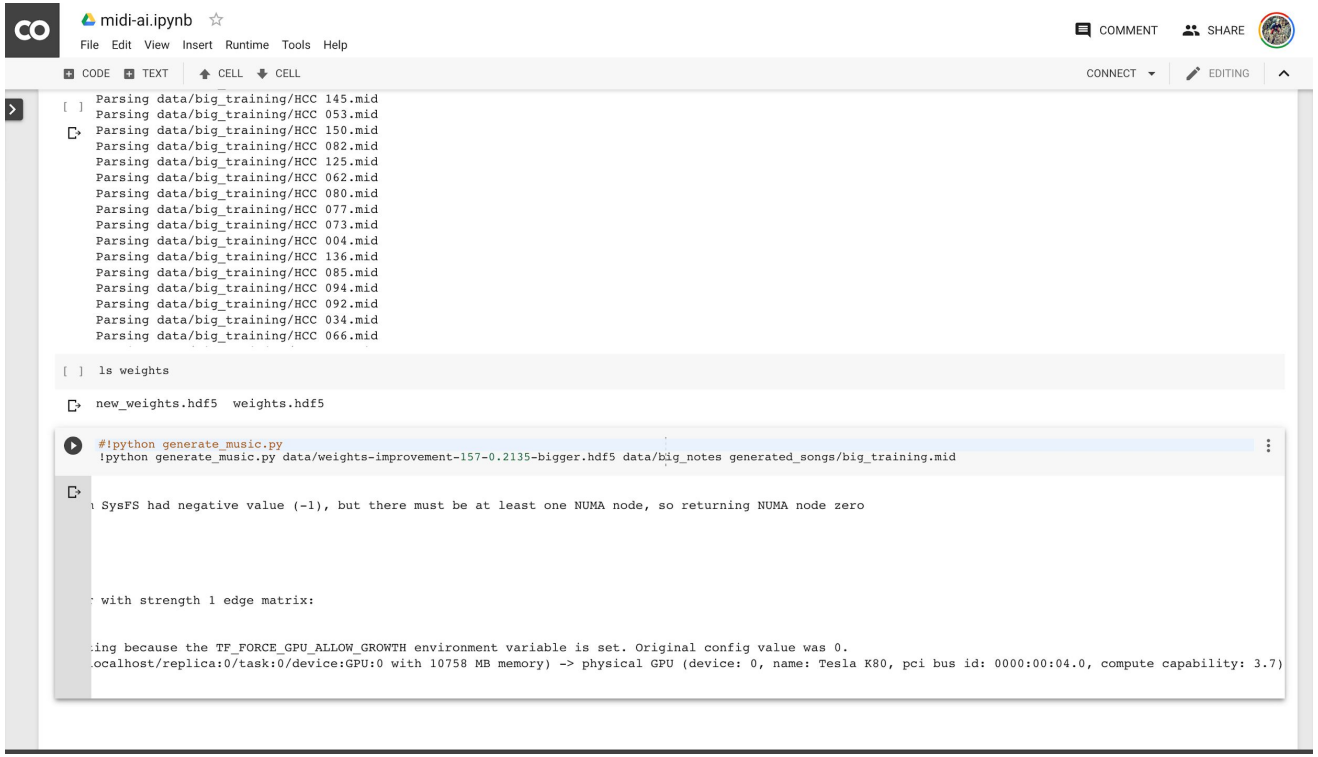
Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.

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



[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

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.Auxiliary')
```

[**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.**

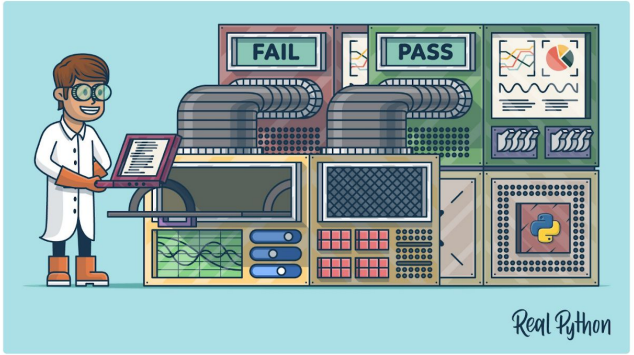
   → [M] Google its benefits ;-)

2. `tail -f filename`

[**SOURCE**]
https://docs.python.org/3/howto/logging-cookbook.html

Getting started with testing in Python

Real Python | Start Here | Tutorials | Products | More | Search...



Getting Started With Testing in Python

by Anthony Shaw | Oct 22, 2018 | 17 Comments | best-practices | intermediate | testing

Table of Contents

— FREE Email Series —

Python Tricks

```
# How to merge two dicts
# in Python 3.5+

>>> x = {'a': 1, 'b': 2}
>>> y = {'b': 3, 'c': 4}

>>> z = {**x, **y}

>>> z
{'c': 4, 'a': 1, 'b': 3}
```

Email...

Get Python Tricks »

No spam. Unsubscribe any time.

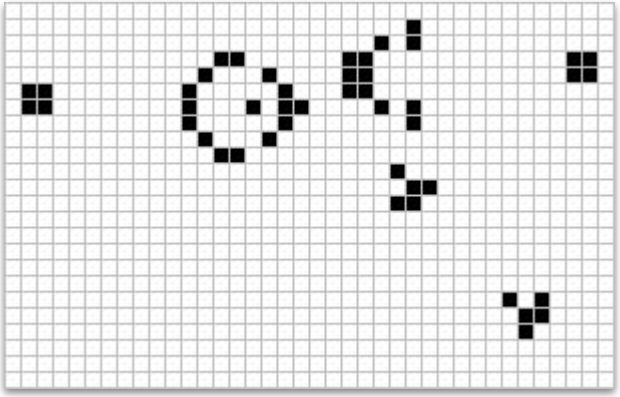
All Tutorial Topics

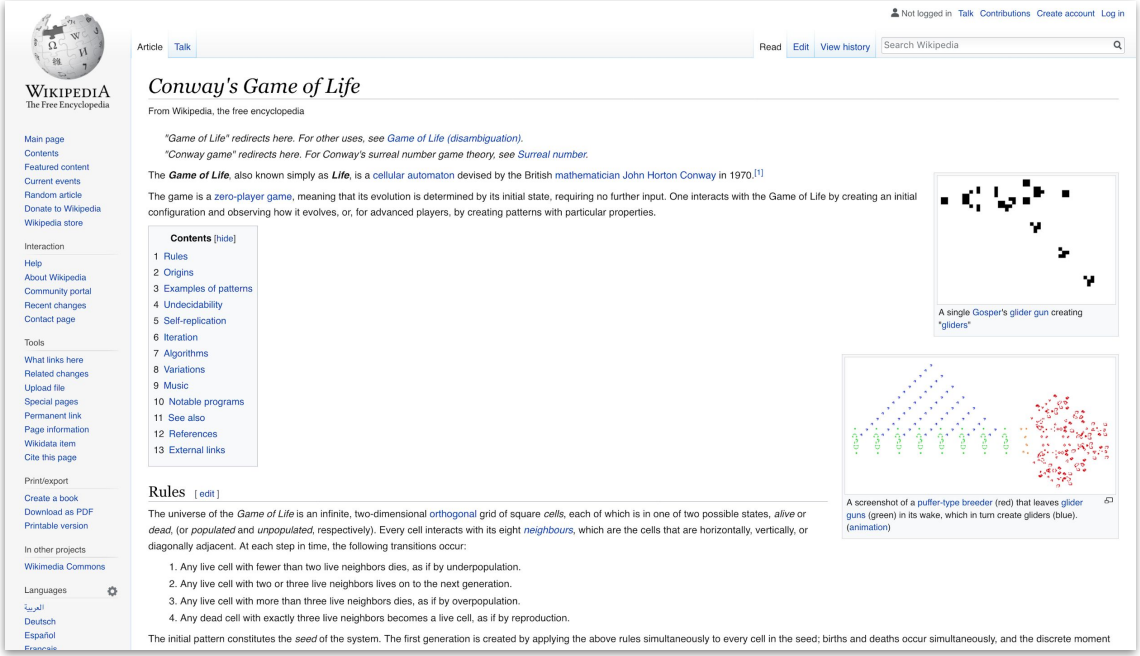
advanced | api | basics | best-practices | community | databases | data-science | devops | django | docker | flask | front-end | intermediate | machine-learning | python | testing | tools | web-dev | web-scraping

Improve Your Python

[SOURCE]
https://realpython.com/python-testing

[R] Conway's game of life





Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

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

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.

A single Gosper's glider gun creating "gliders"

Contents [hide]

1 Rules
2 Origins
3 Examples of patterns
4 Undecidability
5 Self-replication
6 Iteration
7 Algorithms
8 Variations
9 Music
10 Notable programs
11 See also
12 References
13 External links

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

Rules [edit]

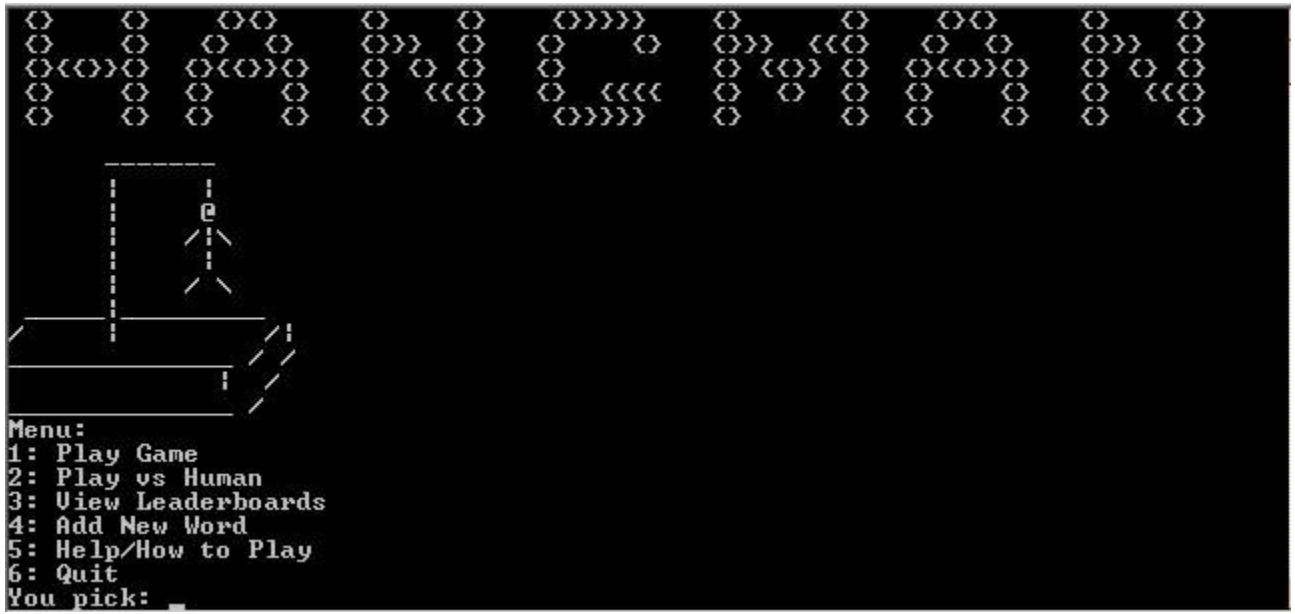
The universe of the *Game of Life* is an infinite, two-dimensional orthogonal grid of square *cells*, each of which is in one of two possible states, *alive* or *dead*, (or *populated* and *unpopulated*, respectively). Every cell interacts with its eight *neighbours*, which are the cells that are horizontally,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

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

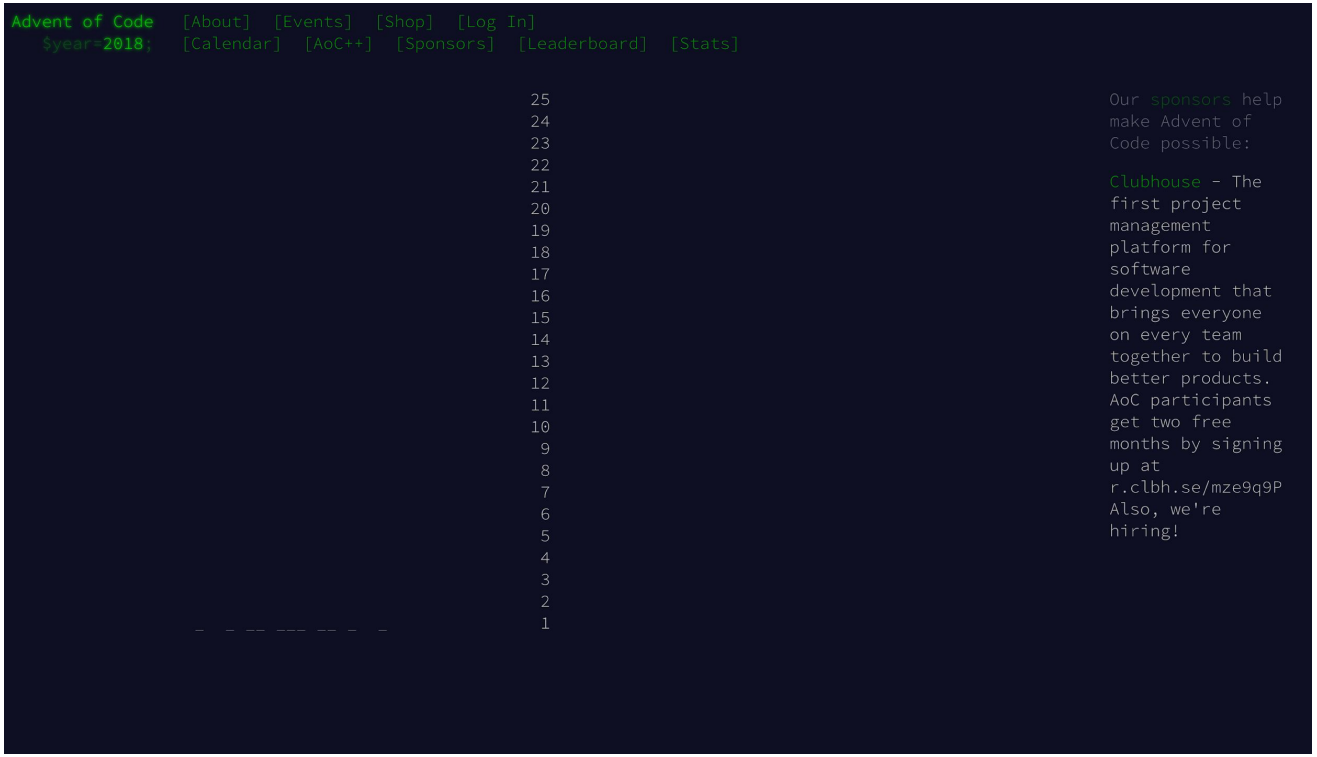
The initial pattern constitutes the *seed* of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed; births and deaths occur simultaneously, and the discrete moment

[**SOURCE**]
https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

[R] Assignments: Hangman



[R → C] Assignments: advent of code



[SOURCE]
https://adventofcode.com

[R] Assignments: Connect 4

[R] Assignments: Stratego

[R] Assignments: Battleship

[F] Assignments: scraper

```python
#!/usr/env python3

import sys
import requests
from bs4 import BeautifulSoup
import os
import time
import re

def get_midi_file_urls(search_term):
    urls      = []
    page      = 1
    next_page = True
    while next_page:
        next_page = False
        url_begin = "http://www.midiworld.com/search/"
        url_end   = "/?q=" + search_term.replace(" ", "%20")

        url      = url_begin + str(page) + url_end
        next_url = url_begin + str(page + 1) + url_end
        response = requests.get(url)
        html = response.text
        soup = BeautifulSoup(html, "html.parser")

        for li in soup.find_all("li"):
            if "download" in str(li):
                for a in li.find_all("a", target="_blank"):
                    urls.append(a.get("href"))

        for link in soup.find_all('a'):
            if link.get("href") == next_url:
                next_page = True
                page = page + 1
                break

        time.sleep(0.01)

    return urls

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: " + sys.argv[0] + " search_term")
        print("   This will query midiworld.com and if there are any results")
        print("   create a directory called \"search_term\" and store all the results in it.")
        exit(-1)

    search_term = sys.argv[1]

    print("Searching...")
```

```python
if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: " + sys.argv[0] + " search_term")
        print("   This will query midiworld.com and if there are any results")
        print("   create a directory called \"search_term\" and store all the results in it.")
        exit(-1)

    search_term = sys.argv[1]

    print("Searching...")
    urls = get_midi_file_urls(search_term)

    if not 0 < len(urls):
        print("No files where found!")
        exit(-1)

    if 0 < len(urls) and not os.path.exists(search_term):
        os.makedirs(search_term)

    print("Downloading...")
    for url in urls:
        tries     = 0
        completed = False
        while not completed:
            try:
                response     = requests.get(url, allow_redirects=True)
                disposition  = response.headers['content-disposition']
                filename     = re.findall("filename=(.+)", disposition)[0] # Only the first occurrence
                print("  Saving " + filename + " into directory " + search_term)
                open(os.path.join(search_term, filename), "wb").write(response.content)
                time.sleep(0.01)
                completed = True
            except:
                tries = tries + 1
                print("Failed to download: " + filename + " (" + str(tries) + " tries)")
                time.sleep(0.1)
                if tries  == 5:
                    completed = True
```

[INFO]

Example http://www.midiworld.com scraper