Biopython Project Update 2019

Standing on each other's shoulders

Peter Cock (@pjacock on Twitter),
The Biopython Contributors (@biopython on Twitter)

The James Hutton Institute

Contents

What is Biopython?

The James Hutton Institute

- Collection of modules for biological computation in Python
  - Sequence handling and motifs, parsers, database queries, protein structures, phylogenetics, tool wrappers and more.
- Started in 1999, first release in 2000
- Open source and freely available (Biopython license)
- `https://biopython.org` and @Biopython on Twitter

biopython

38 named contributors in last year, 16 newcomers with star!

- Alona Levy-Jurgenson*
- Andrey Raspopov*
- Antony Lee
- Ariel Aptekmann
- Benjamin Rowell*
- Bernhard Thiel
- Brandon Invergo
- Catherine Lesuisse
- Chris Rands
- Darcy Mason*
- Deepak Khatri*
- Devang Thakkar*
- Gert Hulselmans
- Ivan Antonov*
- Jared Andrews
- Jens Thomas*
- Jeremy LaBarage*
- Juraj Szász*
- Kai Blin
- Konstantin Vdovkin*
- Lenna Peterson
- Manuel Nuno Melo*
- Mark Amery
- Markus Piotrowski
- Maximilian Greil
- Micky Yun Chan*
- Nick Negretti*
- Peter Cock
- Peter Kerpedjiev
- Ralf Stephan
- Rob Miller
- Rona Costello*
- Sergio Valqui
- Spencer Bliven
- Victor Lin
- Wibowo 'Bow' Arindrarto
- Yi Hsiao*
- Zheng Ruan

Recent releases

- Biopython 1.73 (April 2019)
  - Added Python 3.7 support
- Biopython 1.74 (July 2019)
  - Finally have full API "docstring" coverage,
    special mention to recent contributor Sergio Valqui
- Most code changes to:
  - API documentation
  - Python coding style
    - No consensus on adopting `black` Python formatting style
  - Dual licensing...

Biopython's Open Source License

- Open Source Initiative `https://opensource.org/` maintains a list of approved open source licenses
- Biopython's license is not (quite) on that list
- We're gradually dual-licensing under 3-clause BSD license
- Requires checking each file to confirm all contributors agree
- Biopython 1.70 (July 2017), about 2% of main code done
- Biopython 1.72 (June 2018), about 30% of main code done
- Biopython 1.74 (July 2019), just over 50% of main code done

Documentation

- Documentation on our main website
  - Originally MediaWiki
  - Converted to Markdown using GitHub Pages
- Tutorial and Cookbook
  - Written in LaTeXfor PDF and HTML
- API documentation (Python "docstrings" in the code)
  - Had been turned into HTML using `epydoc`
  - Standardised on reStructuredText (RST)
  - Now using Sphinx `apidoc`

Documentation - biopython.org

The James Hutton Institute

biopython.org

biopython

Python Tools for Computational Molecular Biology

- Documentation
- Download
- Mailing lists
- News
- Biopython Contributors
- Scriptcentral
- Source Code
- GitHub project

Biopython version 1.74
© 2019. All rights reserved.

Edit this page on GitHub

Biopython

See also our News feed and Twitter.

Introduction

Biopython is a set of freely available tools for biological computation written in Python by an international team of developers.

It is a distributed collaborative effort to develop Python libraries and applications which address the needs of current and future work in bioinformatics. The source code is made available under the Biopython License, which is extremely liberal and compatible with almost every license in the world.

We are a member project of the Open Bioinformatics Foundation (OBF), who take care of our domain name and hosting for our mailing list etc. The OBF used to host our developement repository, issue tracker and website but these are now on GitHub.

This page will help you download and install Biopython, and start using the libraries and tools.

Get Started	Get help	Contribute
Download Biopython	Tutorial (PDF)	What's being worked on
Main README	Documentation on this wiki	Developing on Github
	Cookbook (working examples)	Google Summer of Code
	Discuss and ask questions	Report bugs

The latest release is Biopython 1.74, released on 16 July 2019.

Documentation - API via epydoc (old)

The James Hutton Institute

biopython.org

36 matches fastaio Done

Table of Contents

Everything

Modules

Bio
Bio.Affy
Bio.Affy.CelFile
Bio.Align
Bio.Align.AlignInfo
Bio.Align.Applications
Bio.Align.Applications._ClustalOr

Everything

All Classes

Bio.Affy.CelFile.ParserError
Bio.Affy.CelFile.Record
Bio.Align.AlignInfo.PSSM
Bio.Align.AlignInfo.SummaryInfo
Bio.Align.Applications._ClustalOr
Bio.Align.Applications._Clustalw.
Bio.Align.Applications._Dialign.D
Bio.Align.Applications._MSAProb
Bio.Align.Applications._Mafft.Ma
Bio.Align.Applications._Muscle.M
Bio.Align.Applications._Prank.Pra
Bio.Align.Applications._Probcons
Bio.Align.Applications._TCoffee.T
Bio.Align.MultipleSeqAlignment
Bio.Align.PairwiseAligner
Bio.Align.PairwiseAlignment
Bio.Align.PairwiseAlignments
Bio.AlignIO.ClustalIO.ClustalItera
Bio.AlignIO.ClustalIO.ClustalWrit
Bio.AlignIO.EmbossIO.EmbossIte
Bio.AlignIO.EmbossIO.EmbossW
Bio.AlignIO.Interfaces.AlignmentI
Bio.AlignIO.Interfaces.Alignment
Bio.AlignIO.Interfaces.Sequential
Bio.AlignIO.MafIO.MafIndex

Trees Indices Help **Biopython**

Package Bio :: Package SeqIO :: Module FastaIO

[hide private]
[frames] | no frames]

Module FastaIO

source code

Bio.SeqIO support for the "fasta" (aka FastA or Pearson) file format.

You are expected to use this module via the Bio.SeqIO functions.

Classes [hide private]

	`FastaWriter` Class to write Fasta format files (OBSOLETE).
	`FastaTwoLineWriter` Class to write 2-line per record Fasta format files (OBSOLETE).

Functions [hide private]

	`SimpleFastaParser(handle)` Iterate over Fasta records as string tuples.	source code
	`FastaTwoLineParser(handle)` Iterate over no-wrapping Fasta records as string tuples.	source code
	`FastaIterator(handle, alphabet=SingleLetterAlphabet(), title2ids=None)` Iterate over Fasta records as SeqRecord objects.	source code
	`FastaTwoLineIterator(handle, alphabet=SingleLetterAlphabet())` Iterate over two-line Fasta records (as SeqRecord objects).	source code
	`as_fasta(record)` Turn a SeqRecord into a FASTA formated string.	source code
	`as_fasta_2line(record)` Turn a SeqRecord into a two-line FASTA formated string.	source code

Variables [hide private]

	`__package__ = 'Bio.SeqIO'`

Function Details [hide private]

SimpleFastaParser(handle) source code

Iterate over Fasta records as string tuples.

For each record a tuple of two strings is returned, the FASTA title line (without the leading '>' character), and the sequence (with any whitespace removed). The title line is not divided up into an identifier (the first word) and comment or description.

Documentation - API via Sphinx apidoc

The James Hutton Institute

biopython.org

biopython

1.75.dev0

Search docs

API CONTENTS:

- Bio package
  - Subpackages
    - Bio.Affy package
    - Bio.Align package
    - Bio.AlignIO package
    - Bio.Alphabet package
    - Bio.Application package
    - Bio.Blast package
    - Bio.CAPS package
    - Bio.Cluster package
    - Bio.Compass package
    - Bio.Crystal package
    - Bio.Data package
    - Bio.Emboss package
    - Bio.Entrez package

Docs » Bio package » Bio.SeqIO package » Bio.SeqIO.FastaIO module

Previous Next

Bio.SeqIO.FastaIO module

Bio.SeqIO support for the “fasta” (aka FastA or Pearson) file format.

You are expected to use this module via the Bio.SeqIO functions.

Bio.SeqIO.FastaIO.FastaIterator(*handle, alphabet=SingleLetterAlphabet(), title2ids=None*)

Iterate over Fasta records as SeqRecord objects.

Arguments:

- handle - input file
- alphabet - optional alphabet
- title2ids - A function that, when given the title of the FASTA file (without the beginning >), will return the id, name and description (in that order) for the record as a tuple of strings. If this is not given, then the entire title line will be used as the description, and the first word as the id and name.

By default this will act like calling Bio.SeqIO.parse(handle, “fasta”) with no custom handling of the title lines:

```
>>> with open("Fasta/dups.fasta") as handle:
...     for record in FastaIterator(handle):
...         print(record.id)
...
alpha
beta
gamma
```

In progress: Tutorial & API via Sphinx?

The James Hutton Institute

biopython.readthedocs.io

biopython

latest

Search docs

BIOPYTHON TUTORIAL & COOKBOOK:

- Introduction
- Quick Start – What can you do with Biopython?
- Sequence objects
- Sequence annotation objects
- Sequence Input/Output
  - Parsing or Reading Sequences
    - Parsing sequences from compressed files
  - Parsing sequences from the net
  - Sequence files as Dictionaries
  - Writing Sequence Files
    - Low level FASTA and FASTQ parsers
- Multiple Sequence Alignment objects
- BLAST

Read the Docs v: latest

Docs » Sequence Input/Output

Edit on GitHub

Previous Next

Sequence Input/Output

In this chapter we'll discuss in more detail the `Bio.SeqIO` module, which was briefly introduced in Chapter Quick Start – What can you do with Biopython? and also used in Chapter Sequence annotation objects. This aims to provide a simple interface for working with assorted sequence file formats in a uniform way. See also the `Bio.SeqIO` wiki page (http://biopython.org/wiki/SeqIO), and the built in documentation (also online):

```
>>> from Bio import SeqIO
>>> help(SeqIO)
...
```

The "catch" is that you have to work with `SeqRecord` objects (see Chapter Sequence annotation objects), which contain a `Seq` object (see Chapter Sequence objects) plus annotation like an identifier and description. Note that when dealing with very large FASTA or FASTQ files, the overhead of working with all these objects can make scripts too slow. In this case consider the low-level `SimpleFastaParser` and `FastqGeneralIterator` parsers which return just a tuple of strings for each record (see Section Low level FASTA and FASTQ parsers).

Parsing or Reading Sequences

The workhorse function `Bio.SeqIO.parse()` is used to read in sequence data as SeqRecord objects. This function expects two arguments:

1. The first argument is a *handle* to read the data from, or a filename. A handle is typically a file

Documentation - Automation

- Website - automatic publishing on GitHub pages
- Tutorial & cookbook - manual step for each release
- Running `epydoc` was manual step for each release
- Running Sphinx `apidoc` under TravisCI
  - Deploying Sphinx `apidoc` via GitHub Pages
  - Automatic destination based on version, e.g.
    - `https://biopython.org/docs/1.74/api/`
    - `https://biopython.org/docs/dev/api/`
- Next steps
  - Fine tune Sphinx `apidoc` presentation
  - Better match Sphinx and website themes
  - Automate Tutorial & Cookbook build and deploy

Release builds - Automation

The James Hutton Institute

- Build pre-compiled wheels on AppVeyor & TravisCI
  - Following NumPy community in using the multibuild system, developed by Matthew Brett and the MacPython project.
    `https://github.com/matthew-brett/multibuild`
    `https://github.com/biopython/biopython-wheels`
- Uploaded to Python Package Index (PyPI)
- Recommend `pip install biopython`
- Thanks to conda-forge, can do `conda install biopython`

Python Versions - Testing Automation

- Currently test on Python 2.7, 3.4, 3.5, 3.6, and 3.7
- About to drop Python 3.4 and add Python 3.8
- Clear end of life for Python 2 support in 2020, we've pledged on `http://python3statement.org/`
- Also support and test on PyPy
- Support for Jython deprecated

On going and planned work

The James Hutton Institute

- Further simplify release & documentation builds
- Improving compliance with PEP8 and PEP257 style guidelines
- Start working towards `numpydoc` style for docstrings?
- Improving code test coverage
  `https://codecov.io/github/biopython/biopython/`
- Removing/simplifying legacy Alphabet objects
- Dropping Python 2 support in early 2020 (see above)
- Other contributor driven efforts

Changes to help Community Building

The James Hutton Institute

- Already use:
  - GitHub Issue templates (helps bug reporting)
  - GitHub Pull Request templates (to help with expectations)
  - *Easy Fix* tag on some issues, intended for new contributors
  - `CONTRIBUTING` file highlighting coding conventions etc
  - `CODEOWNERS` file to help assign code reviews
- Discussing an OBF-wide *Code of Conduct*
- **What else should we be doing?**

Acknowledgements

Thank you to:

- All our contributors to date
- Contributors' funders for indirect support
- Google Summer of Code (supporting past students)
- Open Bioinformatics Foundation (OBF) for domain name, mailing lists, etc – any maybe later a Code of Conduct?