06_Pandas

January 19, 2025

1 Pandas

The numpy module is excellent for numerical computations, but to handle missing data or arrays with mixed types takes more work. The pandas module is currently the most widely used tool for data manipulation, providing high-performance, easy-to-use data structures and advanced data analysis tools.

In particular pandas features:

- A fast and efficient "DataFrame" object for data manipulation with integrated indexing;
- Tools for reading and writing data between in-memory data structures and different formats (CSV, Excel, SQL, HDF5);
- Intelligent data alignment and integrated handling of missing data;
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets;
- Aggregating or transforming data with a powerful **group-by** engine;
- High performance merging and joining of data sets;
- Hierarchical axis indexing provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure;
- Time series-functionalities;
- Highly optimized for performance, with critical code paths written in C-Python or C.

[54]:
```
import pandas as pd
import numpy as np

! pip show pandas # show information about the library, including version
```

```
Name: pandas
Version: 2.2.3
Summary: Powerful data structures for data analysis, time series, and statistics
Home-page: https://pandas.pydata.org
Author:
Author-email: The Pandas Development Team <pandas-dev@python.org>
License: BSD 3-Clause License

Copyright (c) 2008-2011, AQR Capital Management, LLC, Lambda Foundry, Inc. and
PyData Development Team
All rights reserved.

Copyright (c) 2011-2023, Open source contributors.
```

ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.Copyright 2017- Paul Ganssle <paul@ganssle.io>
Copyright 2017- dateutil contributors (see AUTHORS file)

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

The above license applies to all contributions after 2017-12-01, as well as
all contributions that have been re-licensed (see AUTHORS file for the list of
contributors who have re-licensed their code).

----

dateutil - Extensions to the standard Python datetime module.

Copyright (c) 2003-2011 - Gustavo Niemeyer <gustavo@niemeyer.net>
Copyright (c) 2012-2014 - Tomi Pieviläinen <tomi.pievilainen@iki.fi>
Copyright (c) 2014-2016 - Yaron de Leeuw <me@jarondl.net>
Copyright (c) 2015- - Paul Ganssle <paul@ganssle.io>
Copyright (c) 2015- - dateutil contributors (see AUTHORS file)

All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice,
  this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.
* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,

PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The above BSD License Applies to all code, even that also covered by Apache 2.0.# MIT License

Copyright (c) 2019 Hadley Wickham; RStudio; and Evan Miller

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Based on http://opensource.org/licenses/MIT

This is a template. Complete and ship as file LICENSE the following 2 lines (only)

YEAR:
COPYRIGHT HOLDER:

and specify as

License: MIT + file LICENSE

Copyright (c) <YEAR>, <COPYRIGHT HOLDER>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
The MIT License

Copyright (c) 2008-   Attractive Chaos <attractor@live.co.uk>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.musl as a whole is licensed under the following standard MIT license:

Copyright © 2005-2020 Rich Felker, et al.

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be

----

Authors/contributors include:

A. Wilcox
Ada Worcester
Alex Dowad
Alex Suykov
Alexander Monakov
Andre McCurdy
Andrew Kelley
Anthony G. Basile
Aric Belsito
Arvid Picciani
Bartosz Brachaczek
Benjamin Peterson
Bobby Bingham
Boris Brezillon
Brent Cook
Chris Spiegel
Clément Vasseur
Daniel Micay
Daniel Sabogal
Daurnimator
David Carlier
David Edelsohn
Denys Vlasenko
Dmitry Ivanov
Dmitry V. Levin
Drew DeVault
Emil Renner Berthing
Fangrui Song
Felix Fietkau
Felix Janda
Gianluca Anzolin
Hauke Mehrtens
He X
Hiltjo Posthuma
Isaac Dunham

Jaydeep Patil
Jens Gustedt
Jeremy Huntwork
Jo-Philipp Wich
Joakim Sindholt
John Spencer
Julien Ramseier
Justin Cormack
Kaarle Ritvanen
Khem Raj
Kylie McClain
Leah Neukirchen
Luca Barbato
Luka Perkov
M Farkas-Dyck (Strake)
Mahesh Bodapati
Markus Wichmann
Masanori Ogino
Michael Clark
Michael Forney
Mikhail Kremnyov
Natanael Copa
Nicholas J. Kain
orc
Pascal Cuoq
Patrick Oppenlander
Petr Hosek
Petr Skocik
Pierre Carrier
Reini Urban
Rich Felker
Richard Pennington
Ryan Fairfax
Samuel Holland
Segev Finer
Shiz
sin
Solar Designer
Stefan Kristiansson
Stefan O'Rear
Szabolcs Nagy
Timo Teräs
Trutz Behn
Valentin Ochs
Will Dietz
William Haddon
William Pitcock

Portions of this software are derived from third-party works licensed under terms compatible with the above MIT license:

The TRE regular expression implementation (src/regex/reg* and src/regex/tre*) is Copyright © 2001-2008 Ville Laurikari and licensed under a 2-clause BSD license (license text in the source files). The included version has been heavily modified by Rich Felker in 2012, in the interests of size, simplicity, and namespace cleanliness.

Much of the math library code (src/math/* and src/complex/*) is
Copyright © 1993,2004 Sun Microsystems or
Copyright © 2003-2011 David Schultz or
Copyright © 2003-2009 Steven G. Kargl or
Copyright © 2003-2009 Bruce D. Evans or
Copyright © 2008 Stephen L. Moshier or
Copyright © 2017-2018 Arm Limited
and labelled as such in comments in the individual source files. All have been licensed under extremely permissive terms.

The ARM memcpy code (src/string/arm/memcpy.S) is Copyright © 2008 The Android Open Source Project and is licensed under a two-clause BSD license. It was taken from Bionic libc, used on Android.

The AArch64 memcpy and memset code (src/string/aarch64/*) are Copyright © 1999-2019, Arm Limited.

The implementation of DES for crypt (src/crypt/crypt_des.c) is Copyright © 1994 David Burren. It is licensed under a BSD license.

The implementation of blowfish crypt (src/crypt/crypt_blowfish.c) was originally written by Solar Designer and placed into the public domain. The code also comes with a fallback permissive license for use in jurisdictions that may not recognize the public domain.

The smoothsort implementation (src/stdlib/qsort.c) is Copyright © 2011 Valentin Ochs and is licensed under an MIT-style license.

The x86_64 port was written by Nicholas J. Kain and is licensed under the standard MIT terms.

The mips and microblaze ports were originally written by Richard Pennington for use in the ellcc project. The original code was adapted by Rich Felker for build system and code conventions during upstream integration. It is licensed under the standard MIT terms.

The mips64 port was contributed by Imagination Technologies and is licensed under the standard MIT terms.

The powerpc port was also originally written by Richard Pennington, and later supplemented and integrated by John Spencer. It is licensed under the standard MIT terms.

All other files which have no copyright comments are original works produced specifically for use as part of this library, written either by Rich Felker, the main author of the library, or by one or more contibutors listed above. Details on authorship of individual files can be found in the git version control history of the project. The omission of copyright and license comments in each file is in the interest of source tree size.

In addition, permission is hereby granted for all public header files (include/* and arch/*/bits/*) and crt files intended to be linked into applications (crt/*, ldso/dlstart.c, and arch/*/crt_arch.h) to omit the copyright notice and permission notice otherwise required by the license, and to use these files without any requirement of attribution. These files include substantial contributions from:

Bobby Bingham
John Spencer
Nicholas J. Kain
Rich Felker
Richard Pennington
Stefan Kristiansson
Szabolcs Nagy

all of whom have explicitly granted such permission.

This file previously contained text expressing a belief that most of the files covered by the above exception were sufficiently trivial not to be subject to copyright, resulting in confusion over whether it negated the permissions granted in the license. In the spirit of permissive licensing, and of not having licensing issues being an obstacle to adoption, that text has been removed.Copyright (c) 2005-2023, NumPy Developers.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided

with the distribution.

* Neither the name of the NumPy Developers nor the names of any contributors may be used to endorse or promote products derived from this software without specific prior written permission.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their

Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

A. HISTORY OF THE SOFTWARE
==========================

Python was created in the early 1990s by Guido van Rossum at Stichting Mathematisch Centrum (CWI, see https://www.cwi.nl) in the Netherlands as a successor of a language called ABC. Guido remains Python's principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for National Research Initiatives (CNRI, see https://www.cnri.reston.va.us) in Reston, Virginia where he released several versions of the software.

In May 2000, Guido and the Python core development team moved to BeOpen.com to form the BeOpen PythonLabs team. In October of the same year, the PythonLabs team moved to Digital Creations, which became Zope Corporation. In 2001, the Python Software Foundation (PSF, see https://www.python.org/psf/) was formed, a non-profit organization created specifically to own Python-related Intellectual Property. Zope Corporation was a sponsoring member of the PSF.

All Python releases are Open Source (see https://opensource.org for the Open Source Definition). Historically, most, but not all, Python releases have also been GPL-compatible; the table below summarizes the various releases.

Release	Derived from	Year	Owner	GPL-compatible? (1)
0.9.0 thru 1.2		1991-1995	CWI	yes
1.3 thru 1.5.2	1.2	1995-1999	CNRI	yes
1.6	1.5.2	2000	CNRI	no
2.0	1.6	2000	BeOpen.com	no
1.6.1	1.6	2001	CNRI	yes (2)
2.1	2.0+1.6.1	2001	PSF	no
2.0.1	2.0+1.6.1	2001	PSF	yes
2.1.1	2.1+2.0.1	2001	PSF	yes
2.1.2	2.1.1	2002	PSF	yes
2.1.3	2.1.2	2002	PSF	yes
2.2 and above	2.1.1	2001-now	PSF	yes

Footnotes:

(1) GPL-compatible doesn't mean that we're distributing Python under the GPL. All Python licenses, unlike the GPL, let you distribute a modified version without making your changes open source. The GPL-compatible licenses make it possible to combine Python with other software that is released under the GPL; the others don't.

(2) According to Richard Stallman, 1.6.1 is not GPL-compatible, because its license has a choice of law clause. According to CNRI, however, Stallman's lawyer has told CNRI's lawyer that 1.6.1 is "not incompatible" with the GPL.

Thanks to the many outside volunteers who have worked under Guido's direction to make these releases possible.

B. TERMS AND CONDITIONS FOR ACCESSING OR OTHERWISE USING PYTHON

```
PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2
--------------------------------------------

1. This LICENSE AGREEMENT is between the Python Software Foundation
("PSF"), and the Individual or Organization ("Licensee") accessing and
otherwise using this software ("Python") in source or binary form and
its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF hereby
grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce,
analyze, test, perform and/or display publicly, prepare derivative works,
distribute, and otherwise use Python alone or in any derivative version,
provided, however, that PSF's License Agreement and PSF's notice of copyright,
i.e., "Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023
Python Software Foundation;
All Rights Reserved" are retained in Python alone or in any derivative version
prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on
or incorporates Python or any part thereof, and wants to make
the derivative work available to others as provided herein, then
Licensee hereby agrees to include in any such work a brief summary of
the changes made to Python.

4. PSF is making Python available to Licensee on an "AS IS"
basis.  PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED.  BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON
FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS
A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON,
OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any
relationship of agency, partnership, or joint venture between PSF and
Licensee.  This License Agreement does not grant permission to use PSF
trademarks or trade name in a trademark sense to endorse or promote
products or services of Licensee, or any third party.
```

8. By copying, installing or otherwise using Python, Licensee agrees to be bound by the terms and conditions of this License Agreement.

BEOPEN.COM LICENSE AGREEMENT FOR PYTHON 2.0

BEOPEN PYTHON OPEN SOURCE LICENSE AGREEMENT VERSION 1

1. This LICENSE AGREEMENT is between BeOpen.com ("BeOpen"), having an office at 160 Saratoga Avenue, Santa Clara, CA 95051, and the Individual or Organization ("Licensee") accessing and otherwise using this software in source or binary form and its associated documentation ("the Software").

2. Subject to the terms and conditions of this BeOpen Python License Agreement, BeOpen hereby grants Licensee a non-exclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use the Software alone or in any derivative version, provided, however, that the BeOpen Python License is retained in the Software, alone or in any derivative version prepared by Licensee.

3. BeOpen is making the Software available to Licensee on an "AS IS" basis. BEOPEN MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, BEOPEN MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

4. BEOPEN SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE SOFTWARE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

5. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

6. This License Agreement shall be governed by and interpreted in all respects by the law of the State of California, excluding conflict of law provisions. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between BeOpen and Licensee. This License Agreement does not grant permission to use BeOpen trademarks or trade names in a trademark sense to endorse or promote products or services of Licensee, or any third party. As an exception, the "BeOpen Python" logos available at http://www.pythonlabs.com/logos.html may be used according to the

permissions granted on that web page.

7. By copying, installing or otherwise using the software, Licensee agrees to be bound by the terms and conditions of this License Agreement.

CNRI LICENSE AGREEMENT FOR PYTHON 1.6.1
----

1. This LICENSE AGREEMENT is between the Corporation for National Research Initiatives, having an office at 1895 Preston White Drive, Reston, VA 20191 ("CNRI"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 1.6.1 software in source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, CNRI hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 1.6.1 alone or in any derivative version, provided, however, that CNRI's License Agreement and CNRI's notice of copyright, i.e., "Copyright (c) 1995-2001 Corporation for National Research Initiatives; All Rights Reserved" are retained in Python 1.6.1 alone or in any derivative version prepared by Licensee. Alternately, in lieu of CNRI's License Agreement, Licensee may substitute the following text (omitting the quotes): "Python 1.6.1 is made available subject to the terms and conditions in CNRI's License Agreement. This Agreement together with Python 1.6.1 may be located on the internet using the following unique, persistent identifier (known as a handle): 1895.22/1013. This Agreement may also be obtained from a proxy server on the internet using the following URL: http://hdl.handle.net/1895.22/1013".

3. In the event Licensee prepares a derivative work that is based on or incorporates Python 1.6.1 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 1.6.1.

4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS" basis. CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS

A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

7. This License Agreement shall be governed by the federal intellectual property law of the United States, including without limitation the federal copyright law, and, to the extent such U.S. federal law does not apply, by the law of the Commonwealth of Virginia, excluding Virginia's conflict of law provisions. Notwithstanding the foregoing, with regard to derivative works based on Python 1.6.1 that incorporate non-separable material that was previously distributed under the GNU General Public License (GPL), the law of the Commonwealth of Virginia shall govern this License Agreement only as to issues arising under or with respect to Paragraphs 4, 5, and 7 of this License Agreement. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between CNRI and Licensee. This License Agreement does not grant permission to use CNRI trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

8. By clicking on the "ACCEPT" button where indicated, or by copying, installing or otherwise using Python 1.6.1, Licensee agrees to be bound by the terms and conditions of this License Agreement.

ACCEPT

CWI LICENSE AGREEMENT FOR PYTHON 0.9.0 THROUGH 1.2

Copyright (c) 1991 - 1995, Stichting Mathematisch Centrum Amsterdam, The Netherlands. All rights reserved.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Stichting Mathematisch Centrum or CWI not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

STICHTING MATHEMATISCH CENTRUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL STICHTING MATHEMATISCH CENTRUM BE LIABLE

FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT
OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

ZERO-CLAUSE BSD LICENSE FOR CODE IN THE PYTHON DOCUMENTATION

Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR
OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.
Copyright (c) 2014, Al Sweigart
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the {organization} nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.Copyright
(c) 2017 Anthony Sottile

Permission is hereby granted, free of charge, to any person obtaining a copy

----

Portions of code from MODP_ASCII - Ascii transformations (upper/lower, etc)
https://github.com/client9/stringencoders

Government shall have only "Restricted Rights" as defined in Clause
252.227-7013 (c) (1) of DFARs. Notwithstanding the foregoing, the
authors grant the U.S. Government and others acting in its behalf
permission to use and distribute the software in accordance with the
terms specified in this license.Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of this License; and
You must cause any modified files to carry prominent notices stating that You

changed the files; and
You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.
You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
```

```
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.
Location:
/Users/miriamzara/LaboratoryOfComputationalPhysics_Y7/myenv/lib/python3.12/site-
packages
Requires: numpy, python-dateutil, pytz, tzdata
Required-by: seaborn
```

1.1 Series

Series are an extention to numpy 1D arrays. The new features are *axis labels* and the possibility to store *heterogeneous* elements. Of paramount importance are the time-series, used to define time evolutions of a phenomenon.

[17]:
```
from string import ascii_lowercase as letters

# Creating a series, accessing indexes, values and values by their index
xs = pd.Series(np.arange(10)*0.5, index=tuple(letters[:10])) # every element of␣
 ↪the series can be labeled !
print ("xs: \n", xs,'\n')
print ("xs indexes: \n",xs.index,'\n')
# Values of the Series are actually a numpy array
print ("xs values: \n", xs.values, '\n')
print("type(xs.values) = " , type(xs.values),'\n')
# To access a single element, two syntaxes are permitted:
print ("f-labeled value: ", xs['f'], ", ", xs.f, '\n')
# To access a subset of the series:
print (xs[['d', 'f', 'h']], '\n')
# the subset is still a pandas series
print (type(xs[['d', 'f', 'h']]), '\n')
```

```
xs:
 a    0.0
b    0.5
c    1.0
d    1.5
e    2.0
f    2.5
g    3.0
h    3.5
i    4.0
j    4.5
dtype: float64

xs indexes:
 Index(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'], dtype='object')
```

```
xs values:
 [0.  0.5 1.  1.5 2.  2.5 3.  3.5 4.  4.5]

type(xs.values) =  <class 'numpy.ndarray'>

f-labeled value:  2.5 ,  2.5

d    1.5
f    2.5
h    3.5
dtype: float64

<class 'pandas.core.series.Series'>
```

[ ]:
```
# Extracting elements and operations: same as numpy array
print (xs[:3],'\n')
print (xs[7:], '\n')
print (xs[::3], '\n')
print (xs[xs>3], '\n')
print (np.exp(xs), '\n')
print (np.mean(xs), np.std(xs), '\n')
```

[18]:
```
# Series can be created from python dictionary too (expectedly: they are so␣
↪similiar!).
# Not that the elements can be whatever!
d = {'b' : 1, 'a' : 'cat', 'c' : [2,3]}
pd.Series(d)
```

[18]:
```
b         1
a       cat
c    [2, 3]
dtype: object
```

A key difference between Series and ndarray is that operations between Series automatically align the data based on label. Thus, you can write computations without considering whether the Series involved have the same labels.

[14]:
```
s = pd.Series(np.random.randn(5), index=tuple(letters[:5]))
print(s)
s = s[1:] + s[:-1] #the first and last indexes are not common among the two␣
↪series, and corresponding operations are simply ignored
print(s) #NaN value is printed where the operation could not be performed
```

```
a   -0.114197
b    1.785389
c   -2.667692
d   -0.278673
e    1.642445
```

```
dtype: float64
a         NaN
b    3.570778
c   -5.335383
d   -0.557346
e         NaN
dtype: float64
```

1.1.1 Time series

Time series are very often used to profile the behaviour of a quantity as a function of time. Pandas has a special index for that, `DatetimeIndex`, that can be created e.g. with the function `pd.data_range()`

[21]:
```
# to define a date, the datetime module is very useful
import datetime as dt

# There are various syntaxes admitted for defining dates:

date_A = dt.date.today()
date_B = dt.datetime(2024,11,27,10,45,10,15)
date_C = 'Nov 27 2024'
date_D = '27/11/2024 10:45:00'

print("dt.date.today(): ", date_A)
print("dt.datetime(2024,11,27,10,45,10,15): ", date_B)

print("'Nov 27 2024': ", date_C) # these are just strings
print("'27/11/2024 10:45:00': ", date_D)

# Create a time series containing dates
days = pd.date_range(date_D, periods=7, freq='D')
print ("days series: ", days)

# Create a time series containing dates
seconds = pd.date_range(date_D, periods=3600, freq='s')
print ("seconds series: ", seconds)
```

```
dt.date.today():  2024-11-27
dt.datetime(2024,11,27,10,45,10,15):  2024-11-27 10:45:10.000015
'Nov 27 2024':  Nov 27 2024
'27/11/2024 10:45:00':  27/11/2024 10:45:00
days series:  DatetimeIndex(['2024-11-27 10:45:00', '2024-11-28 10:45:00',
               '2024-11-29 10:45:00', '2024-11-30 10:45:00',
               '2024-12-01 10:45:00', '2024-12-02 10:45:00',
               '2024-12-03 10:45:00'],
              dtype='datetime64[ns]', freq='D')
seconds series:  DatetimeIndex(['2024-11-27 10:45:00', '2024-11-27 10:45:01',
```

```
               '2024-11-27 10:45:02', '2024-11-27 10:45:03',
               '2024-11-27 10:45:04', '2024-11-27 10:45:05',
               '2024-11-27 10:45:06', '2024-11-27 10:45:07',
               '2024-11-27 10:45:08', '2024-11-27 10:45:09',
               …
               '2024-11-27 11:44:50', '2024-11-27 11:44:51',
               '2024-11-27 11:44:52', '2024-11-27 11:44:53',
               '2024-11-27 11:44:54', '2024-11-27 11:44:55',
               '2024-11-27 11:44:56', '2024-11-27 11:44:57',
               '2024-11-27 11:44:58', '2024-11-27 11:44:59'],
              dtype='datetime64[ns]', length=3600, freq='s')
```

To learn more about the frequency strings, please see this link

1.1.2 Timestamps

Timestamped data is the most basic type of time series data that associates values with points in time. For pandas objects it means using the points in time.

functions like `pd.to_datetime` can be used, for instance, when reading information as string from a dataset.

Timestamp is the pandas equivalent of python's Datetime and is interchangeable with it in most cases.

[ ]:
```
tstamp = pd.Timestamp(dt.datetime(2020, 11, 9))

# internally it counts the nanoseconds from January 1st 19
#tstamp = pd.Timestamp(dt.datetime(1970, 1, 1, 0, 0, 0, 1))
print(tstamp.value)

# when creating a timestamp the format can be explicitly passed
ts = pd.to_datetime('2010/11/12', format='%Y/%m/%d')
print (type(ts))
print (ts.value)
ts = pd.to_datetime('12-11-2010 00:00', format='%d-%m-%Y %H:%M')
print (ts)
print (ts.value)
```

A standard series can be created and (range of) elements can be used as indexes

[ ]:
```
tseries = pd.Series(np.random.normal(10, 1, len(days)), index=days)
# Extracting elements
print (tseries[0:4], '\n')
print (tseries['2024-11-27':'2024-12-03'], '\n') # Note - includes end time
```

`pd.to_datetime` can also be used to create a `DatetimeIndex`:

[ ]:
```
pd.to_datetime([1, 2, 3, 4], unit='D', origin=pd.Timestamp('1980-02-03'))
```

1.2 DataFrame

Basic informations:

- A pandas DataFrame is like a simple tabular spreadsheet.
- For future reference (or for people already familiar with R), a pandas DataFrame is very similar to the R DataFrame.
- Each column in a DataFrame is a Series object.
- The element can be whatever, missing data are dealt with as NaN.

1.2.1 DataFrame creation

A DataFrame can be created implicitly, i.e. by providing the index (row names) and the values stored in a `numpy nd.array`.

In the following example, the index is a DatatimeIndex object.

[22]:
```
entries=10
dates=pd.date_range('11/27/2024 10:45:00',freq='h', periods=entries) # 10 rows␣
↪== 10 measures
df = pd.DataFrame(np.random.randn(entries,4), index=dates,␣
↪columns=['A','B','C','D']) # four variables (e.g. pressure, volume,␣
↪temperature, humidity)
df
```

[22]:
```
                            A         B         C         D
2024-11-27 10:45:00 -0.464981 -0.714608  1.039424 -0.806507
2024-11-27 11:45:00 -1.769178  0.380838 -0.279008 -0.641365
2024-11-27 12:45:00  0.348361  0.208374 -0.503578 -1.706411
2024-11-27 13:45:00 -0.467313  0.321762  0.855027 -0.234357
2024-11-27 14:45:00  0.123160 -1.204769  2.099445 -0.265960
2024-11-27 15:45:00 -0.330046 -0.324374  1.275094 -1.308550
2024-11-27 16:45:00  0.138386 -0.524972 -1.039995  0.147370
2024-11-27 17:45:00  0.663835  0.026001  1.490875  0.402071
2024-11-27 18:45:00 -0.464545 -0.252956  0.384504  0.816329
2024-11-27 19:45:00 -0.022338  1.243782  0.818534  0.536484
```

or by means of a dictionary:

careful: all arrays must have the same length

[28]:
```
df2 = pd.DataFrame(
    { 'A' : 1.,
      'B' : pd.Timestamp('20130102'),
      'C' : pd.Series(1,index=range(4),dtype='float32'),
      'D' : np.arange(7,11),
      #'D' : np.arange(7,10),
      'E' : pd.Categorical(["test","train","test","train"]),
    }
```

```
    )
print(df2)

# check what happens if D and E had different lenghts
# Answer:
# ValueError: All arrays must be of the same length
```

```
     A          B    C   D      E
0  1.0 2013-01-02  1.0   7   test
1  1.0 2013-01-02  1.0   8  train
2  1.0 2013-01-02  1.0   9   test
3  1.0 2013-01-02  1.0  10  train
```

1.2.2 Viewing Data

[ ]:
```
df.head(2)
```

[ ]:
```
df.tail(4)
```

[ ]:
```
df.index
```

[ ]:
```
df.columns
```

[ ]:
```
df.values
```

[ ]:
```
df.describe() # THIS IS VERY NICE !!!!!
```

[ ]:

	A	B	C	D
count	10.000000	10.000000	10.000000	10.000000
mean	-0.224466	-0.084092	0.614032	-0.306090
std	0.665191	0.681166	0.973438	0.817332
min	-1.769178	-1.204769	-1.039995	-1.706411
25%	-0.464872	-0.474823	-0.113130	-0.765221
50%	-0.176192	-0.113478	0.836780	-0.250159
75%	0.134580	0.293415	1.216177	0.338395
max	0.663835	1.243782	2.099445	0.816329

Pay attention: doing the transpose is very inefficient in real world datasets, because most of the times you have a lot more rows (measures, or data points) than values. So, handle with care.

[ ]:
```
df.T
```

[31]:
```
df.sort_index(axis=0,ascending=True) # from the smallest to the biggest
df.sort_index(axis=0,ascending=False) # from the biggest to the smallest
```

[31]:

	A	B	C	D
2024-11-27 19:45:00	-0.022338	1.243782	0.818534	0.536484
2024-11-27 18:45:00	-0.464545	-0.252956	0.384504	0.816329

```
2024-11-27 17:45:00  0.663835  0.026001  1.490875  0.402071
2024-11-27 16:45:00  0.138386 -0.524972 -1.039995  0.147370
2024-11-27 15:45:00 -0.330046 -0.324374  1.275094 -1.308550
2024-11-27 14:45:00  0.123160 -1.204769  2.099445 -0.265960
2024-11-27 13:45:00 -0.467313  0.321762  0.855027 -0.234357
2024-11-27 12:45:00  0.348361  0.208374 -0.503578 -1.706411
2024-11-27 11:45:00 -1.769178  0.380838 -0.279008 -0.641365
2024-11-27 10:45:00 -0.464981 -0.714608  1.039424 -0.806507
```

[34]:
```
df.sort_values(by="C") # re-arrange the rows, but this time basing on the␣
↪values in column C. Default is ascending =True
df.sort_values(by="C", ascending= False)
```

[34]:
```
                            A         B         C         D
2024-11-27 14:45:00  0.123160 -1.204769  2.099445 -0.265960
2024-11-27 17:45:00  0.663835  0.026001  1.490875  0.402071
2024-11-27 15:45:00 -0.330046 -0.324374  1.275094 -1.308550
2024-11-27 10:45:00 -0.464981 -0.714608  1.039424 -0.806507
2024-11-27 13:45:00 -0.467313  0.321762  0.855027 -0.234357
2024-11-27 19:45:00 -0.022338  1.243782  0.818534  0.536484
2024-11-27 18:45:00 -0.464545 -0.252956  0.384504  0.816329
2024-11-27 11:45:00 -1.769178  0.380838 -0.279008 -0.641365
2024-11-27 12:45:00  0.348361  0.208374 -0.503578 -1.706411
2024-11-27 16:45:00  0.138386 -0.524972 -1.039995  0.147370
```

1.3 Selection

1.3.1 Getting slices

The following show how to get part of the DataFrame (i.e. not just the elements)

[37]:
```
# Selecting columns:

## standard and safe
some_column = df['A']
## equivalent but dangerous (imagine blank spaces in the name of the column..)
some_column = df.A

# Selecting rows:

## by counting
print (df[0:3], end= '\n\n')
## or by index
print (df["2024-11-27 10:45:00":"2024-11-27 12:45:00"])
```

```
                            A         B         C         D
2024-11-27 10:45:00 -0.464981 -0.714608  1.039424 -0.806507
2024-11-27 11:45:00 -1.769178  0.380838 -0.279008 -0.641365
2024-11-27 12:45:00  0.348361  0.208374 -0.503578 -1.706411
```

```
                            A         B         C         D
2024-11-27 10:45:00 -0.464981 -0.714608  1.039424 -0.806507
2024-11-27 11:45:00 -1.769178  0.380838 -0.279008 -0.641365
2024-11-27 12:45:00  0.348361  0.208374 -0.503578 -1.706411
```

1.3.2 Selection by label

[ ]:
```
# getting a cross section (part of the DataFrame) using a label
df.loc[dates[0]] # select the row with specified index
```

[ ]:
```
A   -0.464981
B   -0.714608
C    1.039424
D   -0.806507
Name: 2024-11-27 10:45:00, dtype: float64
```

[ ]:
```
# selecting on a multi-axis by label:
df.loc[:,['A','B']]
#a=df.loc[:,['A','B']]
```

[ ]:
```
# showing label slicing, both endpoints are included:
df.loc['2024-11-27 14:45:00':'2024-11-27 16:45:00',['A','B']]
```

`.at` and `.loc` are equivalent methods

[ ]:
```
# getting an individual element
print (df.loc[dates[1],'A'])

# equivalently
print (df.at[dates[1],'A'])
```

1.3.3 Selecting by position

[ ]:
```
# select via the position of the passed integers:
print (df.iloc[3],'\n')

# notation similar to numpy/python
print (df.iloc[3:5,0:2])
```

[ ]:
```
# selecting raws 1,2 and 4 for columns 0 and 2
df.iloc[[1,2,4],[0,2]]
```

[ ]:
```
# slicing rows explicitly
print (df.iloc[1:3,:],'\n')

# slicing columns explicitly
print (df.iloc[:,1:3])
```

```
# selecting an individual element by position
print(df.iloc[1,1])
```

1.3.4 Boolean index

Very powerful way of filtering out data with certain features. Notation is very similar to numpy arrays.

[43]:
```
# Filter by a boolean condition on the values of a single column
df[df['B'] > 0]

# In this case, the rows not meeting the conditions are cutted out. The
↪resulting dataframe is smaller.
```

[43]:
```
                            A         B         C         D
2024-11-27 11:45:00 -1.769178  0.380838 -0.279008 -0.641365
2024-11-27 12:45:00  0.348361  0.208374 -0.503578 -1.706411
2024-11-27 13:45:00 -0.467313  0.321762  0.855027 -0.234357
2024-11-27 17:45:00  0.663835  0.026001  1.490875  0.402071
2024-11-27 19:45:00 -0.022338  1.243782  0.818534  0.536484
```

[46]:
```
# Selecting on the basis of boolean conditions applied to the whole DataFrame
df[df>0]

# In this other case, DataFrame with the same shape is returned, with NaN's
↪where condition is not met.
# Typically you don't want this output, so you need again to filter out the
↪rows with NaN
```

[46]:
```
                            A         B         C         D
2024-11-27 10:45:00       NaN       NaN  1.039424       NaN
2024-11-27 11:45:00       NaN  0.380838       NaN       NaN
2024-11-27 12:45:00  0.348361  0.208374       NaN       NaN
2024-11-27 13:45:00       NaN  0.321762  0.855027       NaN
2024-11-27 14:45:00  0.123160       NaN  2.099445       NaN
2024-11-27 15:45:00       NaN       NaN  1.275094       NaN
2024-11-27 16:45:00  0.138386       NaN       NaN  0.147370
2024-11-27 17:45:00  0.663835  0.026001  1.490875  0.402071
2024-11-27 18:45:00       NaN       NaN  0.384504  0.816329
2024-11-27 19:45:00       NaN  1.243782  0.818534  0.536484
```

1.3.5 Setting

Combination of selection and setting of values

[48]:
```
# setting values by label (same as by position)
df.at[dates[0],'A'] = 0
```

```
# setting and assigning a numpy array
df.loc[:,'D'] = np.array([5] * len(df))

# Defining a brand new column

# method 1
df['E'] = np.arange(len(df))*0.5
# method 2: by means of a pd.Series. CAREFUL: indexes must be the same!
df['E prime'] = pd.Series(np.arange(len(df))*2, index=df.index)

df
```

[48]:

```
                            A         B         C    D    E  E prime
2024-11-27 10:45:00  0.000000 -0.714608  1.039424  5.0  0.0        0
2024-11-27 11:45:00 -1.769178  0.380838 -0.279008  5.0  0.5        2
2024-11-27 12:45:00  0.348361  0.208374 -0.503578  5.0  1.0        4
2024-11-27 13:45:00 -0.467313  0.321762  0.855027  5.0  1.5        6
2024-11-27 14:45:00  0.123160 -1.204769  2.099445  5.0  2.0        8
2024-11-27 15:45:00 -0.330046 -0.324374  1.275094  5.0  2.5       10
2024-11-27 16:45:00  0.138386 -0.524972 -1.039995  5.0  3.0       12
2024-11-27 17:45:00  0.663835  0.026001  1.490875  5.0  3.5       14
2024-11-27 18:45:00 -0.464545 -0.252956  0.384504  5.0  4.0       16
2024-11-27 19:45:00 -0.022338  1.243782  0.818534  5.0  4.5       18
```

[ ]:

```
def dcos(theta):
    theta = theta*(np.pi/180)
    return np.cos(theta)

df['cosine'] = pd.Series(df["E"].apply(dcos), index=df.index)
df
```

[ ]:

```
# another example of global setting
df2=df.copy()

df2[df2>0] = -df2
df2
```

1.3.6 Are you dealing with a Copy or a View?

In general is hard to tell. There is no real rule. See the following example to see how tricky it is:

Both actions are performed with the same method, `.loc`. Just calling the method in two slightly different ways produces a copy in one case and a view in the other!

[51]:

```
dfd = pd.DataFrame({'a': [1, 2, 3], 'b': [4, 5, 6]})
print (dfd, end= '\n\n')
```

```python
# This is likely a view
subset = dfd.loc[0:1, 'a']
subset[0] = 100  # May affect `df`

print (dfd, end= '\n\n')

# This is a copy
subset = dfd.loc[[0, 1], 'a']
subset[0] = 200  # Does NOT affect `df`

print (dfd)
```

```
   a  b
0  1  4
1  2  5
2  3  6

     a  b
0  100  4
1    2  5
2    3  6

     a  b
0  100  4
1    2  5
2    3  6
```

The behaviour depend on the version of Pandas and on the version of Numpy that given version of Pandas depends upon.

Since Pandas 1.5 "Copy-on-Write" (CoW) is (optionally) available and as of Pandas 3.0 will be the default.

With CoW, **chained assignemt will never work**.

In the following example, the view `df["foo"]` and `df` itself are modified in one step. This will lead to a `ChainedAssignemntError`

[52]:
```python
dfd["a"][dfd["b"] > 5] = 100
dfd
```

```
/var/folders/vk/kftm8379123bsmwrdp8l0xr00000gn/T/ipykernel_3337/449088826.py:1:
FutureWarning: ChainedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in
certain cases, but when using Copy-on-Write (which will become the default
behaviour in pandas 3.0) this will never work to update the original DataFrame
or Series, because the intermediate object on which we are setting values will
behave as a copy.
A typical example is when you are setting values in a column of a DataFrame,
like:
```

```
df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in
a single step and ensure this keeps updating the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

  dfd["a"][dfd["b"] > 5] = 100
```

[52]:

	a	b
0	100	4
1	2	5
2	100	6

With Copy On Write, the chained operation can be substituted by using `loc`:

[ ]:
```
dfd.loc[dfd["b"] > 5, "a"] = 200
dfd
```

1.3.7 Dropping

N.B.: dropping doesn't act permanently on the DataFrame, i.e. to get that do :

`df = df.drop(....)`

[ ]:
```
# Dropping by column
df.drop(['E prime'], axis=1)

#which is equivalent to
new_df=df.drop(columns=['E prime'])
new_df
```

[ ]:
```
# Dropping by raws
# safe and always working
df.drop(df.index[[1,2,3,4]])
```

[ ]:
```
df
```

[ ]:
```
# something like df.drop('index_name')
# would work but the type of index must be specified,
# in particular with DatetimeIndex
df.drop(pd.to_datetime("2024-11-27 18:45:00"))
```

1.4 Missing data

pandas primarily uses the value `np.nan` to represent missing data. It is by default not included in computations.

[55]:
```
df_wNan = df[df>0]
df_wNan
```

[55]:
```
                            A         B         C    D    E  E prime
2024-11-27 10:45:00       NaN       NaN  1.039424  5.0  NaN      NaN
2024-11-27 11:45:00       NaN  0.380838       NaN  5.0  0.5      2.0
2024-11-27 12:45:00  0.348361  0.208374       NaN  5.0  1.0      4.0
2024-11-27 13:45:00       NaN  0.321762  0.855027  5.0  1.5      6.0
2024-11-27 14:45:00  0.123160       NaN  2.099445  5.0  2.0      8.0
2024-11-27 15:45:00       NaN       NaN  1.275094  5.0  2.5     10.0
2024-11-27 16:45:00  0.138386       NaN       NaN  5.0  3.0     12.0
2024-11-27 17:45:00  0.663835  0.026001  1.490875  5.0  3.5     14.0
2024-11-27 18:45:00       NaN       NaN  0.384504  5.0  4.0     16.0
2024-11-27 19:45:00       NaN  1.243782  0.818534  5.0  4.5     18.0
```

[56]:
```
# dropping rows with at least a Nan
df_wNan.dropna(how='any') # drop if any (at least one) of the elements is a NaN
```

[56]:
```
                            A         B         C    D    E  E prime
2024-11-27 17:45:00  0.663835  0.026001  1.490875  5.0  3.5     14.0
```

[61]:
```
# getting a mask
df_wNan.isna()
#df_wNan.notna()
```

[61]:
```
                         A      B      C      D      E  E prime
2024-11-27 10:45:00   True   True  False  False   True     True
2024-11-27 11:45:00   True  False   True  False  False    False
2024-11-27 12:45:00  False  False   True  False  False    False
2024-11-27 13:45:00   True  False  False  False  False    False
2024-11-27 14:45:00  False   True  False  False  False    False
2024-11-27 15:45:00   True   True  False  False  False    False
2024-11-27 16:45:00  False   True   True  False  False    False
2024-11-27 17:45:00  False  False  False  False  False    False
2024-11-27 18:45:00   True   True  False  False  False    False
2024-11-27 19:45:00   True  False  False  False  False    False
```

[ ]:
```
# filling missing data
#
# (use with care or not at all when dealing with real datasets)
# you might forget that you filled it and interpret the points as valid later
 ↪on in the analysis
#
df_wNan.fillna(value=0)
```

[ ]:
```
                            A         B         C    D    E  E prime
2024-11-27 10:45:00  0.000000  0.000000  1.039424  5.0  0.0      0.0
2024-11-27 11:45:00  0.000000  0.380838  0.000000  5.0  0.5      2.0
```

```
2024-11-27 12:45:00  0.348361  0.208374  0.000000  5.0  1.0      4.0
2024-11-27 13:45:00  0.000000  0.321762  0.855027  5.0  1.5      6.0
2024-11-27 14:45:00  0.123160  0.000000  2.099445  5.0  2.0      8.0
2024-11-27 15:45:00  0.000000  0.000000  1.275094  5.0  2.5     10.0
2024-11-27 16:45:00  0.138386  0.000000  0.000000  5.0  3.0     12.0
2024-11-27 17:45:00  0.663835  0.026001  1.490875  5.0  3.5     14.0
2024-11-27 18:45:00  0.000000  0.000000  0.384504  5.0  4.0     16.0
2024-11-27 19:45:00  0.000000  1.243782  0.818534  5.0  4.5     18.0
```

[65]:
```python
df_wNan.fillna(method= 'pad')
# use this when you think that an interpolation of missing points is reasonable␣
↪enough (data is very regular)
```

```
/var/folders/vk/kftm8379123bsmwrdp8l0xr00000gn/T/ipykernel_3337/2170565733.py:1:
FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a
future version. Use obj.ffill() or obj.bfill() instead.
  df_wNan.fillna(method= 'pad')
```

[65]:
```
                            A         B         C    D    E  E prime
2024-11-27 10:45:00       NaN       NaN  1.039424  5.0  NaN      NaN
2024-11-27 11:45:00       NaN  0.380838  1.039424  5.0  0.5      2.0
2024-11-27 12:45:00  0.348361  0.208374  1.039424  5.0  1.0      4.0
2024-11-27 13:45:00  0.348361  0.321762  0.855027  5.0  1.5      6.0
2024-11-27 14:45:00  0.123160  0.321762  2.099445  5.0  2.0      8.0
2024-11-27 15:45:00  0.123160  0.321762  1.275094  5.0  2.5     10.0
2024-11-27 16:45:00  0.138386  0.321762  1.275094  5.0  3.0     12.0
2024-11-27 17:45:00  0.663835  0.026001  1.490875  5.0  3.5     14.0
2024-11-27 18:45:00  0.663835  0.026001  0.384504  5.0  4.0     16.0
2024-11-27 19:45:00  0.663835  1.243782  0.818534  5.0  4.5     18.0
```

1.5 Operations

Here comes the most relevant advantage of DataFrame. Operations on columns are extremly fast due to several intrinsic optimizations:

- They are implemented in C/Cython via NumPy.
- Pandas processes columns as contiguous memory arrays.
- Vectorized operations eliminate the need for slow Python loops.
- Efficient memory and cache utilization boost performance.

[67]:
```python
# Some statistics (mean() just as an example)
# rows
print (df.mean(axis=0),'\n\n')
# columns
print (df.mean(axis=1),'\n\n')
```

```
A         -0.177968
B         -0.084092
C          0.614032
```

```
D           5.000000
E           2.250000
E prime     9.000000
dtype: float64
```

```
2024-11-27 10:45:00    0.887469
2024-11-27 11:45:00    0.972108
2024-11-27 12:45:00    1.675526
2024-11-27 13:45:00    2.201579
2024-11-27 14:45:00    2.669639
2024-11-27 15:45:00    3.020112
2024-11-27 16:45:00    3.095570
2024-11-27 17:45:00    4.113452
2024-11-27 18:45:00    4.111167
2024-11-27 19:45:00    4.923330
Freq: h, dtype: float64
```

[70]:
```
# global operations on columns

# column values are replaced with their cumulative value, summed from row zero
↪to current row
df.apply(np.cumsum)
```

[70]:

	A	B	C	D	E	E prime
2024-11-27 10:45:00	0.000000	-0.714608	1.039424	5.0	0.0	0
2024-11-27 11:45:00	-1.769178	-0.333771	0.760415	10.0	0.5	2
2024-11-27 12:45:00	-1.420817	-0.125397	0.256837	15.0	1.5	6
2024-11-27 13:45:00	-1.888130	0.196366	1.111864	20.0	3.0	12
2024-11-27 14:45:00	-1.764971	-1.008404	3.211309	25.0	5.0	20
2024-11-27 15:45:00	-2.095017	-1.332778	4.486403	30.0	7.5	30
2024-11-27 16:45:00	-1.956630	-1.857750	3.446408	35.0	10.5	42
2024-11-27 17:45:00	-1.292795	-1.831749	4.937282	40.0	14.0	56
2024-11-27 18:45:00	-1.757340	-2.084705	5.321787	45.0	18.0	72
2024-11-27 19:45:00	-1.779678	-0.840923	6.140320	50.0	22.5	90

[71]:
```
df
```

[71]:

	A	B	C	D	E	E prime
2024-11-27 10:45:00	0.000000	-0.714608	1.039424	5.0	0.0	0
2024-11-27 11:45:00	-1.769178	0.380838	-0.279008	5.0	0.5	2
2024-11-27 12:45:00	0.348361	0.208374	-0.503578	5.0	1.0	4
2024-11-27 13:45:00	-0.467313	0.321762	0.855027	5.0	1.5	6
2024-11-27 14:45:00	0.123160	-1.204769	2.099445	5.0	2.0	8
2024-11-27 15:45:00	-0.330046	-0.324374	1.275094	5.0	2.5	10
2024-11-27 16:45:00	0.138386	-0.524972	-1.039995	5.0	3.0	12

```
2024-11-27 17:45:00  0.663835  0.026001  1.490875  5.0  3.5       14
2024-11-27 18:45:00 -0.464545 -0.252956  0.384504  5.0  4.0       16
2024-11-27 19:45:00 -0.022338  1.243782  0.818534  5.0  4.5       18
```

[72]:
```python
df.apply(lambda x: x.max() - x.min())
```

[72]:
```
A           2.433014
B           2.448552
C           3.139440
D           0.000000
E           4.500000
E prime    18.000000
dtype: float64
```

[ ]:
```python
# syntax is as usual similar to that of numpy arrays
df['A'] + df['B']
```

Let's play it hard and load (in memory) a (relatively) large dataset

[ ]:
```python
TODO
# WARNING! link in past notebook was wrong!, (if needed) get the right file from:
!curl -O https://www.dropbox.com/scl/fi/pkkpoxlm7beasryexpdf8/data_000637.txt?rlkey=rkm2em1v57hewglzelmin21c9&e=1&st=v2mipkl4
#https://www.dropbox.com/s/xvjzaxzz3ysphme/data_000637.txt
file_name="data_000637.txt"
data=pd.read_csv(file_name)
data.tail(10)
```

```
zsh:1: parse error near `&'
```

[ ]:
```
Empty DataFrame
Columns: [<a href="https://www.dropbox.com/scl/fi/pkkpoxlm7beasryexpdf8/data_000
637.txt?rlkey=rkm2em1v57hewglzelmin21c9">Found</a>.]
Index: []
```

Let's now do some operations among (elements of) columns

[ ]:
```python
# the one-liner killing it all
data['timens']=data['TDC_MEAS']*25/30+data['BX_COUNTER']*25
```

[ ]:
```python
data['timens']
```

[ ]:
```python
# the old slooow way
def conversion(data):
    result=[]
    for i in range(len(data)):
        result.append(data.loc[data.index[i],'TDC_MEAS']*25/30.+data.loc[data.index[i],'BX_COUNTER']*25)
```

```python
    return result

data['timens']=conversion(data)
```

Keep in mind: For tasks on extremely large datasets, Pandas is **not** the best option anymore. Nowadays libraries like Polars or Dask can offer even faster alternatives by further parallelizing or optimizing columnar operations.

1.6 Merge

pandas provides various facilities for easily combining together Series, DataFrame, and Panel objects with various kinds of set logic for the indexes and relational algebra functionality in the case of join / merge-type operations.

1.6.1 Concat

concatenation (adding rows) is straightforward

[78]:
```python
rdf = pd.DataFrame(np.random.randn(10, 4))
rdf
```

[78]:
```
          0         1         2         3
0  0.208229  0.277515  0.421068 -0.853007
1 -0.198942  0.279535  0.317665  0.252755
2 -0.606631  0.355541  0.667718 -1.105942
3 -0.328293  1.934024  0.411077 -0.206878
4 -0.432930 -0.438391 -0.514822  0.526117
5  0.028471 -1.644022  0.684910 -0.443443
6 -0.962949 -0.260390  1.303164 -1.168408
7 -0.634376 -0.669610 -0.095975  1.577698
8 -0.547591 -0.509914  0.587067  0.315817
9 -0.028378 -0.602655 -0.403147  0.804114
```

[79]:
```python
# divide it into pieces row-wise
pieces = [rdf[:3], rdf[3:7], rdf[7:]]
pieces
```

[79]:
```
[          0         1         2         3
 0  0.208229  0.277515  0.421068 -0.853007
 1 -0.198942  0.279535  0.317665  0.252755
 2 -0.606631  0.355541  0.667718 -1.105942,
           0         1         2         3
 3 -0.328293  1.934024  0.411077 -0.206878
 4 -0.432930 -0.438391 -0.514822  0.526117
 5  0.028471 -1.644022  0.684910 -0.443443
 6 -0.962949 -0.260390  1.303164 -1.168408,
           0         1         2         3
 7 -0.634376 -0.669610 -0.095975  1.577698
 8 -0.547591 -0.509914  0.587067  0.315817
```

```
9 -0.028378 -0.602655 -0.403147  0.804114]
```

[80]:
```
# put it back together
#pd.concat(pieces)

# indexes can be ignored
pd.concat(pieces, ignore_index=True)

# in case of dimension mismatch, Nan are added where needed
```

[80]:
```
          0         1         2         3
0  0.208229  0.277515  0.421068 -0.853007
1 -0.198942  0.279535  0.317665  0.252755
2 -0.606631  0.355541  0.667718 -1.105942
3 -0.328293  1.934024  0.411077 -0.206878
4 -0.432930 -0.438391 -0.514822  0.526117
5  0.028471 -1.644022  0.684910 -0.443443
6 -0.962949 -0.260390  1.303164 -1.168408
7 -0.634376 -0.669610 -0.095975  1.577698
8 -0.547591 -0.509914  0.587067  0.315817
9 -0.028378 -0.602655 -0.403147  0.804114
```

[ ]:
```
# appending a single row (as a Series)
s = rdf.iloc[3]
rdf = pd.concat([rdf,s.to_frame().T], ignore_index=True)
rdf
```

1.6.2 Merge/Join

SQL like operations on table can be performed on DataFrames. This is all rather sophisticated, refer to the doc for more info/examples.

Let's see the various merging options with the following two example dataframes

[82]:
```
df1 = pd.DataFrame({'id': [1, 2, 3], 'name': ['Alice', 'Bob', 'Charlie']})
df2 = pd.DataFrame({'id': [2, 3, 4], 'age': [25, 30, 35]})
```

Merging, Inner Join (default) Only rows with matching id values are included:

[83]:
```
# Merge on the 'id' column
result = pd.merge(df1, df2, on='id')
print(result)
```

```
   id     name  age
0   2      Bob   25
1   3  Charlie   30
```

Merging, Left Join A left join includes all rows from `df1` (left) and fills in `NaN` for missing matches in `df2`.

[84]:
```
result = pd.merge(df1, df2, on='id', how='left')
print(result)
```

```
   id     name   age
0   1    Alice   NaN
1   2      Bob  25.0
2   3  Charlie  30.0
```

Merging, Outer Join An outer join includes all rows from both DataFrames, filling NaN for missing values

[85]:
```
result = pd.merge(df1, df2, on='id', how='outer')
print(result)
```

```
   id     name   age
0   1    Alice   NaN
1   2      Bob  25.0
2   3  Charlie  30.0
3   4      NaN  35.0
```

`join` is similar to `merge` but uses index as key and has 'Left' as default

[ ]:
```
df1 = pd.DataFrame({'name': ['Alice', 'Bob', 'Charlie']}, index=[1, 2, 3])
df2 = pd.DataFrame({'age': [25, 30, 35]}, index=[2, 3, 4])

result = df1.join(df2)
print(result)
```

1.7 Grouping

By "group by" we are referring to a process involving one or more of the following steps:

- Splitting the data into groups based on some criteria
- Applying a function to each group independently
- Combining the results into a data structure

Grouping is one of the most powerful and at the same time most sofisticated action you can perform with DataFrames. Mastering it is key for an effective usage of Pandas and vectorized data analysis. Reading the documentation or going through a tutorial is **warmly recommended**.

Let's go through a few examples:

[2]:
```
import pandas as pd
data = {
    'Category': ['A', 'B', 'A', 'B', 'A', 'C'],
    'Values': [10, 20, 30, 40, 50, 60]
}

df = pd.DataFrame(data)
```

```python
# Group by 'Category' and calculate the sum
result = df.groupby('Category').sum()
print(result)
```

```
          Values
Category
A             90
B             60
C             60
```

[87]:
```python
# Multiple Aggregations
result = df.groupby('Category').agg(['sum', 'mean'])
print(result)
```

```
         Values
            sum  mean
Category
A            90  30.0
B            60  30.0
C            60  60.0
```

[88]:
```python
# Grouping by multiple columns
df = pd.DataFrame({'Category': ['A', 'A', 'B', 'B', 'C', 'C'],
                   'Type': ['X', 'Y', 'X', 'Y', 'X', 'Y'],
                   'Values': [10, 20, 30, 40, 50, 60]})

result = df.groupby(['Category', 'Type']).sum()
print(result)
```

```
               Values
Category Type
A        X         10
         Y         20
B        X         30
         Y         40
C        X         50
         Y         60
```

[89]:
```python
# Trasformations using groupby(): add group averages to DataFrame
df['Group_Avg'] = df.groupby('Category')['Values'].transform('mean')
print(df)
```

```
  Category Type  Values  Group_Avg
0        A    X      10       15.0
1        A    Y      20       15.0
2        B    X      30       35.0
3        B    Y      40       35.0
4        C    X      50       55.0
```

```
5         C    Y      60        55.0
```

[91]:
```
# filtering
filtered = df.groupby('Category').filter(lambda x: x['Values'].sum() > 50)
print(filtered)
```

```
  Category  Values
0        A      10
1        B      20
2        A      30
3        B      40
4        A      50
5        C      60
```

[ ]:
```
# custom aggregation with apply()
def custom_aggregation(group):
    return pd.Series({
        'Sum': group['Values'].sum(),
        'Max': group['Values'].max(),
        'Count': group['Values'].count()
    })

result = df.groupby('Category').apply(custom_aggregation)
print(result)
```

[ ]:
```
# splitting data into groups
grouped = df.groupby('Category')

for name, group in grouped:
    print(f"Group: {name}")
    print(group)
```

1.8 Multi-indexing

Hierarchical / Multi-level indexing allows sophisticated data analysis on higher dimensional data. In essence, it enables you to store and manipulate data with an arbitrary number of dimensions in lower dimensional data structures like Series (1d) and DataFrame (2d).

[92]:
```
tuples = list(zip(['bar', 'bar', 'baz', 'baz', 'foo', 'foo', 'qux', 'qux'],
          ['one', 'two', 'one', 'two', 'one', 'two', 'one', 'two']))
multi_index = pd.MultiIndex.from_tuples(tuples, names=['first', 'second'])
print (multi_index,'\n')

s = pd.Series(np.random.randn(8), index=multi_index)
print (s)
```

```
MultiIndex([('bar', 'one'),
            ('bar', 'two'),
            ('baz', 'one'),
```

```
            ('baz', 'two'),
            ('foo', 'one'),
            ('foo', 'two'),
            ('qux', 'one'),
            ('qux', 'two')],
           names=['first', 'second'])
```

```
first  second
bar    one      -0.946261
       two       0.145200
baz    one       0.949948
       two       1.062019
foo    one      -1.184764
       two       0.244838
qux    one      -0.647094
       two       0.095144
dtype: float64
```

[ ]:
```
gdf = pd.DataFrame({'A' : ['foo', 'bar', 'foo', 'bar',
                           'foo', 'bar', 'foo', 'foo'],
                    'B' : ['one', 'one', 'two', 'three',
                            'two', 'two', 'one', 'three'],
                    'C' : np.random.randn(8),
                    'D' : np.random.randn(8)})
gdf

# it enables further features of the groupby method,
# e.g. when group-by by multiple columns
gdf.groupby(['A','B']).sum()
```

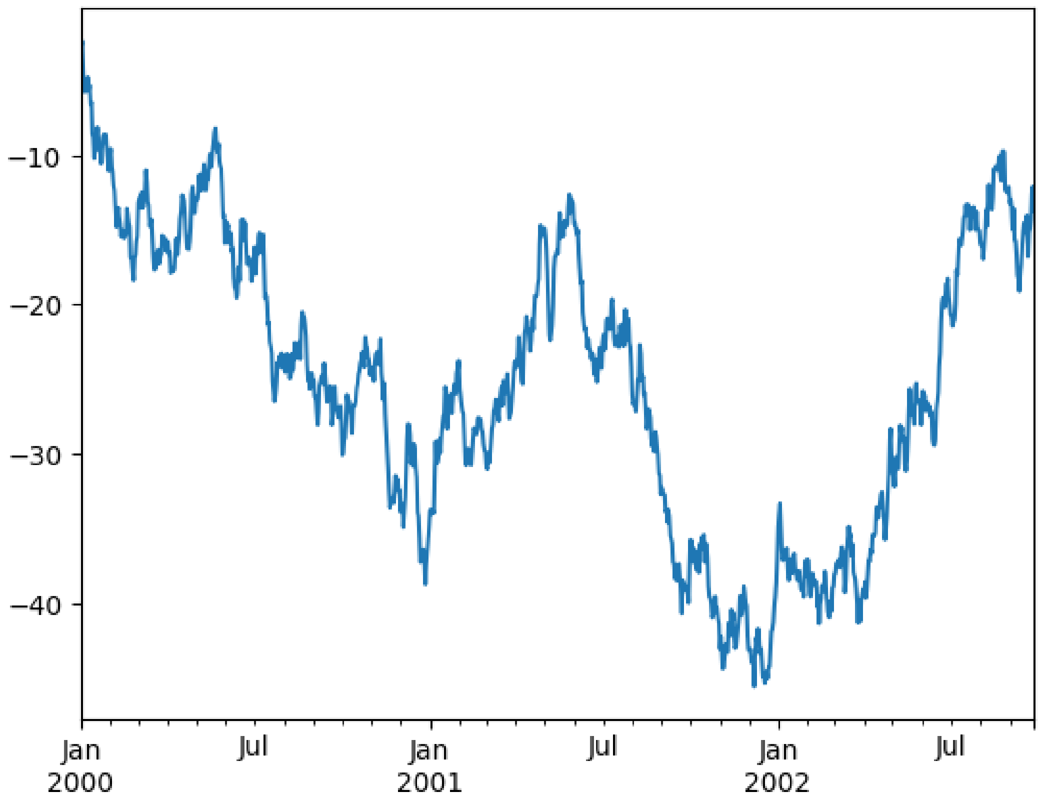
[ ]:
```
# stack() method "compresses" a level in the DataFrame's columns
gdf.groupby(['A','B']).sum().stack()
```

1.9 Plotting

Just a preview, more on the next lab class!

[104]:
```
ts = pd.Series(np.random.randn(1000), index=pd.date_range('1/1/2000',␣
 ↪periods=1000))
ts.cumsum().plot()
```

[104]: <Axes: >

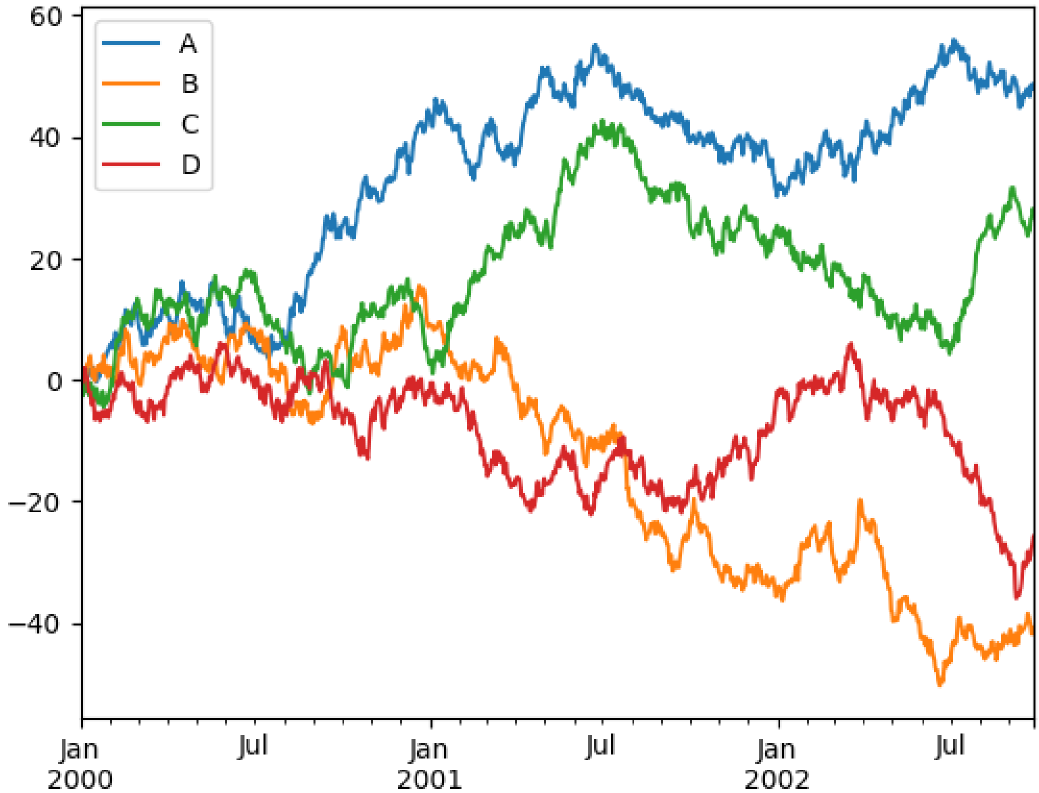


[100]:
```
import matplotlib.pyplot as plt

pdf=pd.DataFrame(np.random.randn(1000, 4), index=ts.index,columns=['A', 'B',␣
↪'C', 'D'])
pdf = pdf.cumsum()
plt.figure(); pdf.plot(); plt.legend(loc='best')
```

[100]:
```
<matplotlib.legend.Legend at 0x1104fc7d0>

<Figure size 640x480 with 0 Axes>
```


60
40
20
0
−20
−40
A
B
C
D
Jan
2000
Jul
Jan
2001
Jul
Jan
2002
Jul