Contributor Covenant Code of Conduct

Our Pledge

In the interest of fostering an open and welcoming environment, we as
contributors and maintainers pledge to making participation in our project and
our community a harassment-free experience for everyone, regardless of age, body
size, disability, ethnicity, sex characteristics, gender identity and expression,
level of experience, education, socio-economic status, nationality, personal
appearance, race, religion, or sexual identity and orientation.

Our Standards

Examples of behavior that contributes to creating a positive environment
include:

* Using welcoming and inclusive language
* Being respectful of differing viewpoints and experiences
* Gracefully accepting constructive criticism
* Focusing on what is best for the community
* Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

* The use of sexualized language or imagery and unwelcome sexual attention or
 advances

* Trolling, insulting/derogatory comments, and personal or political attacks

* Public or private harassment

* Publishing others' private information, such as a physical or electronic
  address, without explicit permission

* Other conduct which could reasonably be considered inappropriate in a
  professional setting

Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be

further defined and clarified by project maintainers.

Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be
reported by contacting the project team at tejas.shetty@iitb.ac.in. All
complaints will be reviewed and investigated and will result in a response that
is deemed necessary and appropriate to the circumstances. The project team is
obligated to maintain confidentiality with regard to the reporter of an incident.
Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good
faith may face temporary or permanent repercussions as determined by other
members of the project's leadership.

Attribution

This Code of Conduct is adapted from the [Contributor Covenant][homepage], version 1.4,
available at https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

[homepage]: https://www.contributor-covenant.org

For answers to common questions about this code of conduct, see
https://www.contributor-covenant.org/faq

Contributing

==========

Contributions are welcome, and they are greatly appreciated! Every little bit

helps, and credit will always be given.

Code of Conduct

Everyone interacting in the ``krotov`` project's code base,

issue tracker, and any communication channels is expected to follow the

`PyPA Code of Conduct`_.

.. _`PyPA Code of Conduct`: https://www.pypa.io/en/latest/code-of-conduct/

Report Bugs

Report bugs at https://github.com/TejasAvinashShetty/repo2pdf/issues.

If you are reporting a bug, please include:

* Your operating system name and version.

* Any details about your local setup that might be helpful in troubleshooting.

* Detailed steps to reproduce the bug.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/TejasAvinashShetty/repo2pdf/issues.

If you are proposing a feature:

* Explain in detail how it would work.
* Keep the scope as narrow as possible, to make it easier to implement.
* Remember that this is a volunteer-driven project, and that contributions
  are welcome :)

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put
   your new functionality into a function with a docstring, and add the
   feature to the list in ``docs/04_features.rst`` and/or ``HISTORY.rst``.

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new

free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of

protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible

feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system

(if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of

copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article

11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product

model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from

a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly

documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is

governed by this License along with a term that is a further
restriction, you may remove that term. If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly
provided under this License. Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)

provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do

not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for

sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to

sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may

not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the

option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation. If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

Later license versions may give you additional or different
permissions. However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify
> it under the terms of the GNU General Public License as published by
> the Free Software Foundation, either version 3 of the License, or
> (at your option) any later version.
>
> This program is distributed in the hope that it will be useful,
> but WITHOUT ANY WARRANTY; without even the implied warranty of
> MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
> GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License
> along with this program. If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License. Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<https://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program
into proprietary programs. If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library. If this is what you want to do, use the GNU Lesser General
Public License instead of this License. But first, please read
<https://www.gnu.org/licenses/why-not-lgpl.html>.

repo2pdf

[![Join the chat at https://gitter.im/repo2pdf/community](https://badges.gitter.im/repo2pdf/community.svg)](https://gitter.im/repo2p

Takes the link to github repository and prints out every file as a PDF

Prerequisites

- Install wkhtml2pdf from https://wkhtmltopdf.org/ or use your package manager

  for example see [Instructions for Ubuntu](https://gist.github.com/brunogaspar/bd89079245923c04be6b0f92af431c10)

Details

Please see [Home](https://github.com/TejasAvinashShetty/repo2pdf/wiki)

References

For references see [references.md](https://github.com/TejasAvinashShetty/repo2pdf/blob/master/references.md)

References

- https://stackabuse.com/python-list-files-in-a-directory/

- https://stackoverflow.com/questions/3207219/how-do-i-list-all-files-of-a-directory#41447012

- https://www.geeksforgeeks.org/packing-and-unpacking-arguments-in-python/

- https://docs.python.org/2/library/os.html#os.listdir

- https://stackoverflow.com/questions/3480184/unpack-a-list-in-python

- https://docs.python.org/3.7/tutorial/controlflow.html#unpacking-argument-lists

I am trying to include all the refrerences. But, I may have missed some.

Security Policy

Supported Versions

Use this section to tell people about which versions of your project are

currently being supported with security updates.

| Version | Supported |

| ------- | ------------------ |

| 5.1.x | :white_check_mark: |

| 5.0.x | :x: |

| 4.0.x | :white_check_mark: |

| < 4.0 | :x: |

Reporting a Vulnerability

Use this section to tell people how to report a vulnerability.

Tell them where to go, how often they can expect to get an update on a

reported vulnerability, what to expect if the vulnerability is accepted or

declined, etc.

---

name: Bug report

about: Create a report to help us improve

title: ''

labels: ''

assignees: ''

Describe the bug

A clear and concise description of what the bug is.

To Reproduce

Steps to reproduce the behavior:

1. Go to '...'

2. Click on '....'

3. Scroll down to '....'

4. See error

Expected behavior

A clear and concise description of what you expected to happen.

Screenshots

If applicable, add screenshots to help explain your problem.

Desktop (please complete the following information):

- OS: [e.g. iOS]

- Browser [e.g. chrome, safari]

- Version [e.g. 22]

Smartphone (please complete the following information):

- Device: [e.g. iPhone6]

- OS: [e.g. iOS8.1]

- Browser [e.g. stock browser, safari]

- Version [e.g. 22]

Additional context

Add any other context about the problem here.

---

name: Custom issue template

about: Describe this issue template's purpose here.

title: ''

labels: ''

assignees: ''

---

name: Feature request

about: Suggest an idea for this project

title: ''

labels: ''

assignees: ''

Is your feature request related to a problem? Please describe.

A clear and concise description of what the problem is. Ex. I'm always frustrated when [...]

Describe the solution you'd like

A clear and concise description of what you want to happen.

Describe alternatives you've considered

A clear and concise description of any alternative solutions or features you've considered.

Additional context

Add any other context or screenshots about the feature request here.

```
name: Greetings

on: [pull_request, issues]

jobs:
  greeting:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/first-interaction@v1
      with:
        repo-token: ${{ secrets.GITHUB_TOKEN }}
        issue-message: 'Message that will be displayed on users'' first issue'
        pr-message: 'Message that will be displayed on users'' first pr'
```

```
# This workflow will triage pull requests and apply a label based on the

# paths that are modified in the pull request.

#

# To use this workflow, you will need to set up a .github/labeler.yml

# file with configuration.  For more information, see:

# https://github.com/actions/labeler/blob/master/README.md


name: Labeler

on: [pull_request]


jobs:

  label:


    runs-on: ubuntu-latest


    steps:

    - uses: actions/labeler@v2

      with:

        repo-token: "${{ secrets.GITHUB_TOKEN }}"
```

```yaml
name: Mark stale issues and pull requests

on:
  schedule:
  - cron: "0 0 * * *"

jobs:
  stale:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/stale@v1
      with:
        repo-token: ${{ secrets.GITHUB_TOKEN }}
        stale-issue-message: 'Stale issue message'
        stale-pr-message: 'Stale pull request message'
        stale-issue-label: 'no-issue-activity'
        stale-pr-label: 'no-pr-activity'
```

```
from os import listdir, getcwd
from os.path import isfile, isdir, join



def folder_opener(folder):
    '''Takes a folder, lists the files and subfolders
    It takes a folder supplied by the user. Then it applies
    listdir (from os module) to get a list of the contents of the
    folder. It then uses isfile and isdir (both from os.path) to sort
    the contents (as obtained above from listdir) to make 2 lists :
    one of files and other one of subfolders.

    Inputs:
    folder : string
            Path to the folder from current working directory

    Outputs:
    list_of_files_in_the_folder: list of strings
                            list containing the strings
                            representing the paths
                            of the files in the folder

    list_of_sub_folders_in_the_folder: list of strings
                                list containing the strings
                                representing the paths
                                of the sub-folders in the folder
```

```
    folder_contents : list of strings
                      list containing the strings representing all the
                      contents of the folder

    '''
    folder_contents = listdir(folder)
    list_of_files_in_the_folder = []
    list_of_sub_folders_in_the_folder = []

    for folder_member in folder_contents:
        folder_member_path = join(folder, folder_member)
        if isfile(folder_member_path) and not isdir(folder_member_path):
            list_of_files_in_the_folder.append(folder_member_path)

        elif isdir(folder_member_path) and not isfile(folder_member_path):
            list_of_sub_folders_in_the_folder.append(folder_member_path)
        else:
            raise ValueError

    return [list_of_files_in_the_folder,
            list_of_sub_folders_in_the_folder,
            folder_contents]


def path_maker(repository_name):
```

'''Makes a list of the path to each repository file

repository_member : constituents of the repository at all
levels subfolders, files,
files of subfolders and so on

Input:

repository_name : str
Name of the repository as a string or
Path to the repo from current working directory

Output:

repository_member_path_list : list of strings
list containing the paths to all
the files (not subfolders) of
the repository.
(files paths are specified
with repect to the head of the
repository)

repo_folder_levelled_dict : dictionary
dictionary containing the
structure of the repository at
level. Keys are levels and values
are the paths to the sub-folders

```
                (not files) at each level.


repo_levelled_dict : dictionary
                dictionary containing the
                structure of the repository at
                level. Keys are levels and values
                are the names of the sub-folders
                and files at each level.
'''
print('Current working directory')
print(getcwd())
repository_member_path_list = []
# repo_folder_levelled_dict = {0: ['.'], }
# repo_levelled_dict = {0: ['.'], }
repo_folder_levelled_dict = {0: [repository_name], }
repo_levelled_dict = {0: [repository_name], }
i = 0


while True:
    folder_list = repo_folder_levelled_dict[i]
    repo_folder_levelled_dict[i + 1] = []
    repo_levelled_dict[i + 1] = []
    for folder in folder_list:
        classified_contents = folder_opener(folder)
        repository_member_path_list.extend(classified_contents[0])
        repo_folder_levelled_dict[i + 1].extend(classified_contents[1])
```

```
            repo_levelled_dict[i + 1].extend(classified_contents[2])


        if repo_folder_levelled_dict[i + 1]:
            i = i + 1
            continue  # i_did_not_open_every_folder
        else:
            break  # If it is empty stop the process
            # since there are no folders


    return [repository_member_path_list,
            repo_folder_levelled_dict,
            repo_levelled_dict]
```

ó

?4¬]c @ sH d d l m Z m Z d d l m Z m Z m Z d „ Z d „ Z d S(iÿÿÿÿ(t listdirt getcwd(t isfilet

It takes a folder supplied by the user. Then it applies

listdir (from os module) to get a list of the contents of the

folder. It then uses isfile and isdir (both from os.path) to sort

the contents (as obtained above from listdir) to make 2 lists :

one of files and other one of subfolders.

Inputs:

folder : string

Path to the folder from current working directory

Outputs:

list_of_files_in_the_folder: list of strings

list containing the strings

representing the paths

of the files in the folder

list_of_sub_folders_in_the_folder: list of strings

list containing the strings

representing the paths

of the sub-folders in the folder

(R R R R t appendt

ValueError(t foldert folder_contentst list_of_files_in_the_foldert! list_of_sub_folders_in_the_foldert

folder_membert folder_member_path((s

path_maker.pyt

folder_opener s

c C së g } i d g d 6} i d g d 6} d } x¯ t rÝ | | } g | | d <g | | d <x] | D]U } t | *f* } | j | d *f* | | d

repository_member : constituents of the repository at all

levels subfolders, files,

files of subfolders and so on

Input:

repository_name : str

Name of the repository as a string or

Path to the repo from current working directory

Output:

repository_member_path_list : list of strings

list containing the paths to all

the files (not subfolders) of

the repository.

(files paths are specified

with repect to the head of the

repository)

repo_folder_levelled_dict : dictionary

dictionary containing the

structure of the repository at

level. Keys are levels and values

are the paths to the sub-folders

(not files) at each level.

t .i i i (t TrueR

t extend(t repository_namet repository_member_path_listt repo_folder_levelled_dictt repository_levelled_dictt it

path_maker.pyt

path_maker0 s(

N(t osR R t os.pathR R R R

R (((s

path_maker.pyt <module> s +

'''

Basically folder names don't end in a file extension.

So one can sort out folders from files.

After git cloning the repository will be present locally.

It will consist of files and folders in the first level.

After this level one will have to recursively go through each folder,

and copy the paths.

'Makefile' is an exception to this line of thinking.

'''

'''

tejas@g3:~/git-tejas/repo2pdf/tests/test-folders/schedule/

schedule-master$ python

Python 2.7.15+ (default, Oct 7 2019, 17:39:04)

[GCC 7.4.0] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>> import os

>>>

>>> for root, dirs, files in os.walk("."):

...   for filename in files:

...   print(filename)

...

test_schedule.py

setup.py

MANIFEST.in

requirements-dev.txt

HISTORY.rst

```
LICENSE.txt

README.rst

FAQ.rst

.gitignore

.travis.yml

AUTHORS.rst

tox.ini

faq.rst

api.rst

index.rst

Makefile

conf.py

placeholder.txt

sidebarintro.html

__init__.py

>>> root

'./schedule'

>>> dirs

[]

>>> os.listdir(".")

['test_schedule.py', 'setup.py', 'MANIFEST.in', 'requirements-dev.txt',

'HISTORY.rst', 'LICENSE.txt', 'docs', 'README.rst', 'FAQ.rst',

'schedule', '.gitignore', '.travis.yml', 'AUTHORS.rst', 'tox.ini']

>>> os.listdir("/schedule")

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>
```

```
OSError: [Errno 2] No such file or directory: '/schedule'
>>> os.listdir("schedule")
['__init__.py']
>>> os.listdir("docs")
['faq.rst', '_static', 'api.rst', 'index.rst', 'Makefile', 'conf.py',
'_templates']
>>> import glob
>>> print(glob.glob("./*"))
['./test_schedule.py', './setup.py', './MANIFEST.in',
'./requirements-dev.txt', './HISTORY.rst', './LICENSE.txt', './docs',
'./README.rst', './FAQ.rst', './schedule', './AUTHORS.rst', './tox.ini']
>>> import os
>>>
>>> def get_filepaths(directory):
...     """
...     This function will generate the file names in a directory
...     tree by walking the tree either top-down or bottom-up.
        For each
...     directory in the tree rooted at directory top (including top
        itself),
...     it yields a 3-tuple (dirpath, dirnames, filenames).
...     """
...     file_paths = []  # List which will store all of the full
        filepaths.
...
>>>     # Walk the tree.
```

```
...     for root, directories, files in os.walk(directory):
  File "<stdin>", line 2
    for root, directories, files in os.walk(directory):
    ^
IndentationError: unexpected indent
>>>         for filename in files:
  File "<stdin>", line 1
    for filename in files:
    ^
IndentationError: unexpected indent
>>>             # Join the two strings in order to form the full
                filepath.
...             filepath = os.path.join(root, filename)
  File "<stdin>", line 2
    filepath = os.path.join(root, filename)
    ^
IndentationError: unexpected indent
>>>             file_paths.append(filepath)  # Add it to the list.
  File "<stdin>", line 1
    file_paths.append(filepath)  # Add it to the list.
    ^
IndentationError: unexpected indent
>>>
>>>     return file_paths  # Self-explanatory.
  File "<stdin>", line 1
    return file_paths  # Self-explanatory.
```

```
    ^
IndentationError: unexpected indent
>>>
>>> # Run the above function and store its results in a vari
...
>>> def files(path):
...     for file in os.listdir(path):
...         if os.path.isfile(os.path.join(path, file)):
...             yield file
...
>>> for file in files("."):
...     print (file)
...
test_schedule.py
setup.py
MANIFEST.in
requirements-dev.txt
HISTORY.rst
LICENSE.txt
README.rst
FAQ.rst
.gitignore
.travis.yml
AUTHORS.rst
tox.ini
>>>
```

```
# Could use isfile and isdir of os.path


Python 3.7.3 (default, Mar 27 2019, 22:11:17)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from os import listdir, getcwd
>>> from os.path import isfile, isdir
>>> getcwd()
'/home/tejas/git-tejas/repo2pdf/tests/test-folders/schedule/schedule-master'
>>> isfile(".")
False
>>> .
  File "<stdin>", line 1
    .
    ^
SyntaxError: invalid syntax
>>> "."
'.'
>>> isdir(".")
True
>>> listdir(".")
['test_schedule.py', 'setup.py', 'MANIFEST.in', 'requirements-dev.txt',
'HISTORY.rst', 'LICENSE.txt', 'docs', 'README.rst', 'FAQ.rst', 'schedule',
 '.gitignore', '.travis.yml', 'AUTHORS.rst', 'tox.ini']
```

```
>>> l = listdir(".")
>>> isfile(l[0])
True
>>> isdir(l[0])
False
>>> l_docs = listdir('docs')
>>> l_docs
['faq.rst', '_static', 'api.rst', 'index.rst', 'Makefile', 'conf.py',
'_templates']
>>> isfile(l_docs[0])
False
>>> isfile('docs/' + l_docs[0])
True
>>> isdir('docs/' + l_docs[0])
False
>>> isdir(l_docs[0])
False
>>> # isfile emits True if it is a file, and false for directory, does not
>>> # exist and every thing else.
>>> # isdir emits True if it is a folder or directory,
>>> # and false for file, does not
>>> # exist and every thing else.
>>> from os.path import join
>>> isfile(join('docs' + l_docs[0]))
False
>>> join('docs' + l_docs[0])
```

```
'docsfaq.rst'

>>> join('docs', l_docs[0])

'docs/faq.rst'

>>> isfile(join('docs',l_docs[0]))

True

>>> isdir(join('docs',l_docs[0]))

False

>>> []

[]

>>> True

True

>>> [] is True

False

>>> ['sd'] is True

False

>>> if ['sd']:

...     print('name')

...

name

>>> if []:

...     print('no name')

...

>>>

>>> if []:

...     print('no name')

... else:
```

```
...     print('has a name')
...
has a name
>>>
>>> repository_folder_levelled_dict = {0: ['.'], }
>>> folder_list = repository_folder_levelled_dict.keys[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'builtin_function_or_method' object is not subscriptable
>>> folder_list = repository_folder_levelled_dict[0]
>>> folder_list
['.']
>>> folder_list1 = repository_folder_levelled_dict[1]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 1
>>> repository_folder_levelled_dict[1] = ['', 'tejas']
>>> folder_list1 = repository_folder_levelled_dict[1]
>>> folder_list1
['', 'tejas']
>>> repository_folder_levelled_dict[1 + 1] = ['', 'tejas', 'shetty']
>>> repository_folder_levelled_dict
{0: ['.'], 1: ['', 'tejas'], 2: ['', 'tejas', 'shetty']}
>>> a = []
>>> a.append(*folder_list1)
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>

TypeError: append() takes exactly one argument (2 given)

>>> a.extend(*folder_list)

>>> a

['.']

>>> a.extend(*folder_list1)

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

TypeError: extend() takes exactly one argument (2 given)

>>> a.extend(folder_list1)

>>> a

['.', '', 'tejas']

>>>

>>> repository_folder_levelled_dict[3] = []

>>> repository_folder_levelled_dict[3].extend(a)

>>> repository_folder_levelled_dict[3]

['.', '', 'tejas']



'''
```

```
from os import mkdir, chdir
from os.path import isfile, isdir
from subprocess import Popen,  PIPE



'''

Ensures a unique 'pdf' directory is ready.
First need to check that 'pdf' directory does not already exist.
Then will make one if it does not exist. Otherwise, will do nothing.
Need to report sucess to user and rest of the program. If fail raise
Folder not found error.
    '''
try:
    # We assume that we are in the folder,
    # where the git repos are stored.
    chdir('../')  # Climb to the upper directory
    if isdir('pdf') and not isfile('pdf'):
        # do nothing
        print('pdf directory exists already.')
    else:
        mkdir('pdf')
        print('pdf directory does not already exist.')
except FileNotFoundError as e: # noqa
    raise FileNotFoundError
else:
```

```
    pass
finally:
    pass



def folder_creator(repo_folder_levelled_dict):
    '''Makes the folder and builds internal structure
    Makes the folder. Also, uses the sub-folder  information
    to build the entire internal structure.
    Assumes we are inside the folder consisting the git
    cloned repository.
    Inputs:

    repo_folder_levelled_dict : dictionary
                        dictionary containing the
                        structure of the repository at
                        level. Keys are levels and values
                        are the paths to the sub-folders
                        (not files) at each level.

    Outputs:
    None
    '''
    chdir('../')
    chdir('pdf')
    for i in repo_folder_levelled_dict.keys():
```

```
        for folder in repo_folder_levelled_dict[i]:
            mkdir(folder)


    return None



def wkhtmltopdf(url, pdf_path):
    '''Interface to wkhtmltopdf
    It takes the url and the pdf_path. Then, it passes it on to
    wkhtmltopdf which then downloads and saves the pdf to the specified
    location.
    Inputs :
    url : str
        URL of the place on the internet from which we must access the
        file.

    pdf_path : str
            Path relative to the current directory where one must
            store (save) the file.

    Output:
    None
    '''
    # process = Popen(["wkhtmltopdf",
    # "https://github.com/TejasAvinashShetty/PH413/blob/master/k_cnot.py",
    # "k_cnot.pdf"])
```

```
process = Popen(["wkhtmltopdf", url, pdf_path])

# process = subprocess.Popen(["ls", "-l"], stdout=subprocess.PIPE)

# (output, err) = process.communicate()



return None
```

'''

Once, the paths are made by path maker,

- make a folder "repo_name_pdf"

- Make a similar folder structure as in the repo

- Use wkhtml2pdf to make the pdfs from each file

- save the resulting pdf file in a simlar way as it's original avatar

"i.e. in the same relative path as the original code file but in the

 path begining from 'repo_name_pdf' instead of 'repo_name'"

-1. Accept the data from path_maker.

 0. Make a folder 'pdf' (make only if it does not already exist.)

    and do all saving of pdfs in 'pdf'

 1. Folder_structure_maker using repo_folder_levelled_dict

 2. Interface to wkhtml2pdf (accept url, path to pdf, name of pdf)

    to produce pdf

 3. Use the above interface

     - to supply url from 'repository_member_path_list'

     - manufacture path to pdf and name from 'repository_member_path_list'

       for example setup.py ---> setup_py.pdf

     - give both of these as above to the interface and we are done

'''

'''

repository_member_dict : dictionary

dictionary containing the

keys as the members

(members at all level subfolders, files,

files of subfolders and so on)

of the repository and the values as their

paths with repect to the head of the

repository

path_list : list of strings

list of path of each file starting from the HEAD of the

repository

for repository_member in

repository_member_dict = {repository_name: '.'}

classified_repo_contents = folder_opener(repository)

repository_member_path_list.append(classified_repo_contents[0])

if classified_repo_contents[1]:

for sub-folders in classified_repo_contents[1]:

else:

break

def test_path_maker():

```
    pass


repo_pdf = repository_name + '_pdf'

mkdir(repo_pdf)

'''
```

```
from os import listdir, getcwd
from os.path import isfile, isdir, join



def folder_opener(folder):
    '''Takes a folder, lists the files and subfolders
    It takes a folder supplied by the user. Then it applies
    listdir (from os module) to get a list of the contents of the
    folder. It then uses isfile and isdir (both from os.path) to sort
    the contents (as obtained above from listdir) to make 2 lists :
    one of files and other one of subfolders.

    Inputs:
    folder : string
            Path to the folder from current working directory

    Outputs:
    list_of_files_in_the_folder: list of strings
                            list containing the strings
                            representing the paths
                            of the files in the folder

    list_of_sub_folders_in_the_folder: list of strings
                                list containing the strings
                                representing the paths
                                of the sub-folders in the folder
```

```
    '''
    folder_contents = listdir(folder)
    list_of_files_in_the_folder = []
    list_of_sub_folders_in_the_folder = []

    for folder_member in folder_contents:
        folder_member_path = join(folder, folder_member)
        if isfile(folder_member_path) and not isdir(folder_member_path):
            list_of_files_in_the_folder.append(folder_member_path)

        elif isdir(folder_member_path) and not isfile(folder_member_path):
            list_of_sub_folders_in_the_folder.append(folder_member_path)
        else:
            raise ValueError

    return [list_of_files_in_the_folder,
            list_of_sub_folders_in_the_folder,
            folder_contents]


def path_maker(repository_name):
    '''Makes a list of the path to each repository file
    repository_member : constituents of the repository at all
                        levels subfolders, files,
                        files of subfolders and so on
```

```
Input:


repository_name : str
                Name of the repository as a string or
                Path to the repo from current working directory


Output:
repository_member_path_list : list of strings
                        list containing the paths to all
                        the files (not subfolders) of
                        the repository.
                        (files paths are specified
                        with repect to the head of the
                        repository)


repo_folder_levelled_dict : dictionary
                            dictionary containing the
                            structure of the repository at
                            level. Keys are levels and values
                            are the paths to the sub-folders
                            (not files) at each level.
'''
repository_member_path_list = []
# repo_folder_levelled_dict = {0: ['.'], }
```

```
# repository_levelled_dict = {0: ['.'], }

repo_folder_levelled_dict = {0: [repository_name], }

repository_levelled_dict = {0: [repository_name], }

i = 0


while True:

    folder_list = repo_folder_levelled_dict[i]

    repo_folder_levelled_dict[i + 1] = []

    repository_levelled_dict[i + 1] = []

    for folder in folder_list:

        classified_contents = folder_opener(folder)

        repository_member_path_list.extend(classified_contents[0])

        repo_folder_levelled_dict[i + 1].extend(classified_contents[1])

        repository_levelled_dict[i + 1].extend(classified_contents[2])


    if repo_folder_levelled_dict[i + 1]:

        i = i + 1

        continue  # i_did_not_open_every_folder

    else:

        break  # If it is empty stop the process

        # since there are no folders


return [repository_member_path_list,

        repo_folder_levelled_dict,

        repository_levelled_dict]
```

ó

8¬]c @ sH d d l m Z m Z d d l m Z m Z m Z d „ Z d „ Z d S(iÿÿÿÿ(t listdirt getcwd(t isfilet

It takes a folder supplied by the user. Then it applies

listdir (from os module) to get a list of the contents of the

folder. It then uses isfile and isdir (both from os.path) to sort

the contents (as obtained above from listdir) to make 2 lists :

one of files and other one of subfolders.

Inputs:

folder : string

Path to the folder from current working directory

Outputs:

list_of_files_in_the_folder: list of strings

list containing the strings

representing the paths

of the files in the folder

list_of_sub_folders_in_the_folder: list of strings

list containing the strings

representing the paths

of the sub-folders in the folder

(R R R R t appendt

ValueError(t foldert folder_contentst list_of_files_in_the_foldert! list_of_sub_folders_in_the_foldert

folder_membert folder_member_path((s2 /home/tejas/git-tejas/repo2pdf/tests/path_maker.pyt

folder_opener s

c C së g } i | g d 6} i | g d 6} d } x¯ t rÝ | | } g | | d <g | | d <x] | D]U } t | ƒ } | j | d ƒ | | d j

repository_member : constituents of the repository at all

levels subfolders, files,

files of subfolders and so on

Input:

repository_name : str

Name of the repository as a string or

Path to the repo from current working directory

Output:

repository_member_path_list : list of strings

list containing the paths to all

the files (not subfolders) of

the repository.

(files paths are specified

with repect to the head of the

repository)

repo_folder_levelled_dict : dictionary

dictionary containing the

structure of the repository at

level. Keys are levels and values

are the paths to the sub-folders

(not files) at each level.

i i i (t TrueR

t extend(t repository_namet repository_member_path_listt repo_folder_levelled_dictt repository_levelled_dictt it

path_maker0 s(

N(t osR R t os.pathR R R R

R (((s2 /home/tejas/git-tejas/repo2pdf/tests/path_maker.pyt <module> s +

'''

['test-folders/schedule/schedule-master/test_schedule.py', 'test-folders/schedule/schedule-master/setup.py', 'test-folders/schedu

()

{0: ['test-folders/schedule'], 1: ['test-folders/schedule/schedule-master'], 2: ['test-folders/schedule/schedule-master/docs', 'test-fol

()

{0: ['test-folders/schedule'], 1: ['schedule-master'], 2: ['test_schedule.py', 'setup.py', 'MANIFEST.in', 'requirements-dev.txt', 'HIST

()

'''

```
# from os import chdir, getcwd, mkdir

from path_maker import path_maker




sch_path = path_maker('test-folders/schedule')

print(sch_path[0])

print()

print(sch_path[1])

print()

print(sch_path[2])

print()

'''

[repository_member_path_list,

        repo_folder_levelled_dict,

        repository_levelled_dict] =




from path_maker import path_maker

from os import chdir, getcwd

chdir('../tests/test-folder')

s_path = path_maker(schedule)

print(s_path[0])

print()

print(s_path[1])

print()

print(s_path[2])
```

```
print()



chdir('../src')

print(getcwd())

from path_maker import path_maker

chdir('../tests')

print(getcwd())

'''
```

<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en-IN"><head><meta content="text/html; cha

</style><style>body,td,a,p,.h{font-family:arial,sans-serif}body{margin:0;overflow-y:scroll}#gog{padding:3px 8px 0}td{line-height:.

if (!iesg){document.f&&document.f.q.focus();document.gbqf&&document.gbqf.q.focus();}

}

})();</script> <nobr>Search <a class=gb1 href="http://www.google.co.in/imghp

else top.location='/doodles/';};})();</script><input value="AAP1E1EAAAAAXas2Gz5JXSXmJ-b3nbiqAgEpmBqeUVQy" name="ifl

function _F_installCss(c){}

(function(){google.spjs=false;google.snet=true;google.em=[];google.emw=false;})();(function(){var pmc='{"CaHQXQ":{},"Qnk92g"

```
*.py[cod]

# C extensions
*.so

# Packages
*.egg
*.egg-info
dist
build
eggs
parts
bin
var
sdist
develop-eggs
.installed.cfg
lib
lib64
MANIFEST

# Installer logs
pip-log.txt

# Unit test / coverage reports
.coverage
```

```
.tox

nosetests.xml


# Translations

*.mo


# Mr Developer

.mr.developer.cfg

.project

.pydevproject


env

env3

__pycache__

venv


.cache

docs/_build

.idea/
```

```
dist: xenial

language: python

python:

    - "2.7"

    - "3.5"

    - "3.6"

    - "3.7"

    - "3.8-dev"

    - "nightly"

install: pip install tox-travis coveralls

script:

    - tox

after_success:

    - coveralls

matrix:

  allow_failures:

    - python: "3.8-dev"

    - python: "nightly"
```

Thanks to all the wonderful folks who have contributed to schedule over the years:

- mattss <https://github.com/mattss>
- mrhwick <https://github.com/mrhwick>
- cfrco <https://github.com/cfrco>
- matrixise <https://github.com/matrixise>
- abultman <https://github.com/abultman>
- mplewis <https://github.com/mplewis>
- WoLfulus <https://github.com/WoLfulus>
- dylwhich <https://github.com/dylwhich>
- fkromer <https://github.com/fkromer>
- alaingilbert <https://github.com/alaingilbert>
- Zerrossetto <https://github.com/Zerrossetto>
- yetingsky <https://github.com/yetingsky>
- schnepp <https://github.com/schnepp> <https://bitbucket.org/saschaschnepp>
- grampajoe <https://github.com/grampajoe>
- gilbsgilbs <https://github.com/gilbsgilbs>
- Nathan Wailes <https://github.com/NathanWailes>
- Connor Skees <https://github.com/ConnorSkees>

.. _frequently-asked-questions:

Frequently Asked Questions

==========================

Frequently asked questions on the usage of ``schedule``.

How to execute jobs in parallel?

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

*I am trying to execute 50 items every 10 seconds, but from the my logs it says it executes every item in 10 second schedule se

By default, schedule executes all jobs serially. The reasoning behind this is that it would be difficult to find a model for parallel ex

You can work around this restriction by running each of the jobs in its own thread:

.. code-block:: python

    import threading
    import time
    import schedule

    def job():
        print("I'm running on thread %s" % threading.current_thread())
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
    def run_threaded(job_func):
        job_thread = threading.Thread(target=job_func)
        job_thread.start()

    schedule.every(10).seconds.do(run_threaded, job)
    schedule.every(10).seconds.do(run_threaded, job)
    schedule.every(10).seconds.do(run_threaded, job)
    schedule.every(10).seconds.do(run_threaded, job)
    schedule.every(10).seconds.do(run_threaded, job)

    while 1:
        schedule.run_pending()
        time.sleep(1)
```

If you want tighter control on the number of threads use a shared jobqueue and one or more worker threads:

```
.. code-block:: python

    import Queue
    import time
    import threading
    import schedule
```

```
def job():
    print("I'm working")



def worker_main():
    while 1:
        job_func = jobqueue.get()
        job_func()
        jobqueue.task_done()

jobqueue = Queue.Queue()

schedule.every(10).seconds.do(jobqueue.put, job)
schedule.every(10).seconds.do(jobqueue.put, job)
schedule.every(10).seconds.do(jobqueue.put, job)
schedule.every(10).seconds.do(jobqueue.put, job)
schedule.every(10).seconds.do(jobqueue.put, job)

worker_thread = threading.Thread(target=worker_main)
worker_thread.start()

while 1:
    schedule.run_pending()
    time.sleep(1)
```

This model also makes sense for a distributed application where the workers are separate processes that receive jobs from a di

How to continuously run the scheduler without blocking the main thread?

~~~~~~

Run the scheduler in a separate thread. Mrwhick wrote up a nice solution in to this problem `here <https://github.com/mrhwick/s

Does schedule support timezones?

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Vanilla schedule doesn't support timezones at the moment. If you need this functionality please check out @imiric's work `here <

What if my task throws an exception?

~~~~~~

Schedule doesn't catch exceptions that happen during job execution. Therefore any exceptions thrown during job execution will

If you want to guard against exceptions you can wrap your job function

in a decorator like this:

.. code-block:: python

```
import functools


def catch_exceptions(cancel_on_failure=False):
    def catch_exceptions_decorator(job_func):
```
~~~~~~

```
        @functools.wraps(job_func)
        def wrapper(*args, **kwargs):
            try:
                return job_func(*args, **kwargs)
            except:
                import traceback
                print(traceback.format_exc())
                if cancel_on_failure:
                    return schedule.CancelJob
        return wrapper
    return catch_exceptions_decorator

@catch_exceptions(cancel_on_failure=True)
def bad_task():
    return 1 / 0

schedule.every(5).minutes.do(bad_task)
```

Another option would be to subclass Schedule like @mplewis did in `this example <https://gist.github.com/mplewis/8483f1c24f2

How can I run a job only once?

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
.. code-block:: python

    def job_that_executes_once():
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
        # Do some work ...
        return schedule.CancelJob

    schedule.every().day.at('22:30').do(job_that_executes_once)
```

How can I cancel several jobs at once?

~~~~~~

You can cancel the scheduling of a group of jobs selecting them by a unique identifier.

.. code-block:: python

```
    def greet(name):
        print('Hello {}'.format(name))

    schedule.every().day.do(greet, 'Andrea').tag('daily-tasks', 'friend')
    schedule.every().hour.do(greet, 'John').tag('hourly-tasks', 'friend')
    schedule.every().hour.do(greet, 'Monica').tag('hourly-tasks', 'customer')
    schedule.every().day.do(greet, 'Derek').tag('daily-tasks', 'guest')

    schedule.clear('daily-tasks')
```

Will prevent every job tagged as ``daily-tasks`` from running again.
~~~~~~

I'm getting an ``AttributeError: 'module' object has no attribute 'every'`` when I try to use schedule. How can I fix this?

~~~~~~

This happens if your code imports the wrong ``schedule`` module. Make sure you don't have a ``schedule.py`` file in your projec

How can I add generic logging to my scheduled jobs?

~~~~~~

The easiest way to add generic logging functionality to your schedule

job functions is to implement a decorator that handles logging

in a reusable way:

.. code-block:: python

```
    import functools
    import time

    import schedule


    # This decorator can be applied to
    def with_logging(func):
        @functools.wraps(func)
        def wrapper(*args, **kwargs):
            print('LOG: Running job "%s"' % func.__name__)
            result = func(*args, **kwargs)
```

```
            print('LOG: Job "%s" completed' % func.__name__)
            return result
        return wrapper

    @with_logging
    def job():
        print('Hello, World.')

    schedule.every(3).seconds.do(job)

    while 1:
        schedule.run_pending()
        time.sleep(1)
```

How to run a job at random intervals?

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
.. code-block:: python

    def my_job():
        # This job will execute every 5 to 10 seconds.
        print('Foo')

    schedule.every(5).to(10).seconds.do(my_job)
```

How can I pass arguments to the job function?
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~

``do()`` passes extra arguments to the job function:

.. code-block:: python

    def greet(name):
        print('Hello', name)

    schedule.every(2).seconds.do(greet, name='Alice')
    schedule.every(4).seconds.do(greet, name='Bob')

How can I make sure long-running jobs are always executed on time?

~~~~~~

Schedule does not account for the time it takes the job function to execute. To guarantee a stable execution schedule you need

.. :changelog:

History

0.6.0 (2019-01-20)

++++++++++++++++++

- Make at() accept timestamps with 1 second precision (#267). Thanks @NathanWailes!

- Introduce proper exception hierarchy (#271). Thanks @ConnorSkees!

0.5.0 (2017-11-16)

++++++++++++++++++

- Keep partially scheduled jobs from breaking the scheduler (#125)

- Add support for random intervals (Thanks @grampajoe and @gilbsgilbs)

0.4.3 (2017-06-10)

++++++++++++++++++

- Improve docs & clean up docstrings

0.4.2 (2016-11-29)

++++++++++++++++++

- Publish to PyPI as a universal (py2/py3) wheel

0.4.0 (2016-11-28)

++++++++++++++++++

- Add proper HTML (Sphinx) docs available at https://schedule.readthedocs.io/

- CI builds now run against Python 2.7 and 3.5 (3.3 and 3.4 should work fine but are untested)

- Fixed an issue with ``run_all()`` and having more than one job that deletes itself in the same iteration. Thanks @alaingilbert.

- Add ability to tag jobs and to cancel jobs by tag. Thanks @Zerrossetto.

- Improve schedule docs. Thanks @Zerrossetto.

- Additional docs fixes by @fkromer and @yetingsky.

0.3.2 (2015-07-02)

++++++++++++++++++

- Fixed issues where scheduling a job with a functools.partial as the job function fails. Thanks @dylwhich.

- Fixed an issue where scheduling a job to run every >= 2 days would cause the initial execution to happen one day early. Thanl

- Added a FAQ item to describe how to schedule a job that runs only once.

0.3.1 (2014-09-03)

++++++++++++++++++

- Fixed an issue with unicode handling in setup.py that was causing trouble on Python 3 and Debian (https://github.com/dbader/s

- Added an FAQ item to describe how to deal with job functions that throw exceptions. Thanks @mplewis.

0.3.0 (2014-06-14)

++++++++++++++++++

- Added support for scheduling jobs on specific weekdays. Example: ``schedule.every().tuesday.do(job)`` or ``schedule.every().w

- Run tests against Python 2.7 and 3.4. Python 3.3 should continue to work but we're not actively testing it on CI anymore.

0.2.1 (2013-11-20)

++++++++++++++++++

- Fixed history (no code changes).

0.2.0 (2013-11-09)

++++++++++++++++++

- This release introduces two new features in a backwards compatible way:

- Allow jobs to cancel repeated execution: Jobs can be cancelled by calling ``schedule.cancel_job()`` or by returning ``schedule.

- Updated ``at_time()`` to allow running jobs at a particular time every hour. Example: ``every().hour.at(':15').do(job)`` will run ``jo

- Refactored unit tests to mock ``datetime`` in a cleaner way. (Thanks @matts.)

0.1.11 (2013-07-30)

+++++++++++++++++++

- Fixed an issue with ``next_run()`` throwing a ``ValueError`` exception when the job queue is empty. Thanks to @dpagano for p

0.1.10 (2013-06-07)

++++++++++++++++++

- Fixed issue with ``at_time`` jobs not running on the same day the job is created (Thanks to @mattss)

0.1.9 (2013-05-27)

++++++++++++++++++

- Added ``schedule.next_run()``

- Added ``schedule.idle_seconds()``

- Args passed into ``do()`` are forwarded to the job function at call time

- Increased test coverage to 100%

0.1.8 (2013-05-21)

++++++++++++++++++

- Changed default ``delay_seconds`` for ``schedule.run_all()`` to 0 (from 60)

- Increased test coverage

0.1.7 (2013-05-20)

++++++++++++++++++

- API change: renamed ``schedule.run_all_jobs()`` to ``schedule.run_all()``

- API change: renamed ``schedule.run_pending_jobs()`` to ``schedule.run_pending()``

- API change: renamed ``schedule.clear_all_jobs()`` to ``schedule.clear()``

- Added ``schedule.jobs``

0.1.6 (2013-05-20)

++++++++++++++++++

- Fix packaging

- README fixes

0.1.4 (2013-05-20)

++++++++++++++++++

- API change: renamed ``schedule.tick()`` to ``schedule.run_pending_jobs()``

- Updated README and ``setup.py`` packaging

0.1.0 (2013-05-19)

++++++++++++++++++

- Initial release

The MIT License (MIT)

```
include README.rst

include HISTORY.rst

include LICENSE.txt


include test_schedule.py


recursive-exclude * __pycache__

recursive-exclude * *.py[co]
```

schedule

========

.. image:: https://api.travis-ci.org/dbader/schedule.svg?branch=master

:target: https://travis-ci.org/dbader/schedule

.. image:: https://coveralls.io/repos/dbader/schedule/badge.svg?branch=master

:target: https://coveralls.io/r/dbader/schedule

.. image:: https://img.shields.io/pypi/v/schedule.svg

:target: https://pypi.python.org/pypi/schedule

Python job scheduling for humans.

An in-process scheduler for periodic jobs that uses the builder pattern

for configuration. Schedule lets you run Python functions (or any other

callable) periodically at pre-determined intervals using a simple,

human-friendly syntax.

Inspired by `Adam Wiggins' <https://github.com/adamwiggins>`_ article `"Rethinking Cron" <https://adam.herokuapp.com/past/2

Features

- A simple to use API for scheduling jobs.

- Very lightweight and no external dependencies.

- Excellent test coverage.

- Tested on Python 2.7, 3.5, and 3.6

Usage

.. code-block:: bash

    $ pip install schedule

.. code-block:: python

    import schedule
    import time

    def job():
        print("I'm working...")

    schedule.every(10).minutes.do(job)
    schedule.every().hour.do(job)
    schedule.every().day.at("10:30").do(job)
    schedule.every(5).to(10).minutes.do(job)
    schedule.every().monday.do(job)
    schedule.every().wednesday.at("13:15").do(job)
    schedule.every().minute.at(":17").do(job)

```
    while True:
        schedule.run_pending()
        time.sleep(1)
```

Documentation

Schedule's documentation lives at `schedule.readthedocs.io <https://schedule.readthedocs.io/>`_.

Please also check the FAQ there with common questions.

Meta

Daniel Bader - `@dbader_org <https://twitter.com/dbader_org>`_ - mail@dbader.org

Distributed under the MIT license. See `LICENSE.txt <https://github.com/dbader/schedule/blob/master/LICENSE.txt>`_ for more

https://github.com/dbader/schedule

docutils

mock

Pygments

pytest

pytest-cov

pytest-flake8

Sphinx

```
"""
Publish a new version:

$ git tag X.Y.Z -m "Release X.Y.Z"
$ git push --tags

$ pip install --upgrade twine wheel
$ python setup.py sdist bdist_wheel --universal
$ twine upload dist/*
"""
import codecs
from setuptools import setup


SCHEDULE_VERSION = '0.6.0'
SCHEDULE_DOWNLOAD_URL = (
    'https://github.com/dbader/schedule/tarball/' + SCHEDULE_VERSION
)


def read_file(filename):
    """
    Read a utf8 encoded text file and return its contents.
    """
    with codecs.open(filename, 'r', 'utf8') as f:
        return f.read()
```

```
setup(
    name='schedule',
    packages=['schedule'],
    version=SCHEDULE_VERSION,
    description='Job scheduling for humans.',
    long_description=read_file('README.rst'),
    license='MIT',
    author='Daniel Bader',
    author_email='mail@dbader.org',
    url='https://github.com/dbader/schedule',
    download_url=SCHEDULE_DOWNLOAD_URL,
    keywords=[
        'schedule', 'periodic', 'jobs', 'scheduling', 'clockwork',
        'cron', 'scheduler', 'job scheduling'
    ],
    classifiers=[
        'Intended Audience :: Developers',
        'License :: OSI Approved :: MIT License',
        'Programming Language :: Python',
        'Programming Language :: Python :: 3',
        'Programming Language :: Python :: 2.7',
        'Programming Language :: Python :: 3.5',
        'Programming Language :: Python :: 3.6',
        'Programming Language :: Python :: 3.7',
```

```
        'Natural Language :: English',
    ],
    python_requires='>=2.7,!=3.0.*,!=3.1.*,!=3.2.*,!=3.3.*,!=3.4.*',
)
```

```
"""Unit tests for schedule.py"""

import datetime

import functools

import mock

import unittest


# Silence "missing docstring", "method could be a function",

# "class already defined", and "too many public methods" messages:

# pylint: disable-msg=R0201,C0111,E0102,R0904,R0901


import schedule

from schedule import every, ScheduleError, ScheduleValueError, IntervalError




def make_mock_job(name=None):

    job = mock.Mock()

    job.__name__ = name or 'job'

    return job




class mock_datetime(object):

    """

    Monkey-patch datetime for predictable results

    """

    def __init__(self, year, month, day, hour, minute, second=0):

        self.year = year
```

```
        self.month = month
        self.day = day
        self.hour = hour
        self.minute = minute
        self.second = second

    def __enter__(self):
        class MockDate(datetime.datetime):
            @classmethod
            def today(cls):
                return cls(self.year, self.month, self.day)

            @classmethod
            def now(cls):
                return cls(self.year, self.month, self.day,
                           self.hour, self.minute, self.second)
        self.original_datetime = datetime.datetime
        datetime.datetime = MockDate

    def __exit__(self, *args, **kwargs):
        datetime.datetime = self.original_datetime


class SchedulerTests(unittest.TestCase):
    def setUp(self):
        schedule.clear()
```

```
def test_time_units(self):
    assert every().seconds.unit == 'seconds'
    assert every().minutes.unit == 'minutes'
    assert every().hours.unit == 'hours'
    assert every().days.unit == 'days'
    assert every().weeks.unit == 'weeks'

    job_instance = schedule.Job(interval=2)
    # without a context manager, it incorrectly raises an error because
    # it is not callable
    with self.assertRaises(IntervalError):
        job_instance.minute
    with self.assertRaises(IntervalError):
        job_instance.hour
    with self.assertRaises(IntervalError):
        job_instance.day
    with self.assertRaises(IntervalError):
        job_instance.week
    with self.assertRaises(IntervalError):
        job_instance.monday
    with self.assertRaises(IntervalError):
        job_instance.tuesday
    with self.assertRaises(IntervalError):
        job_instance.wednesday
    with self.assertRaises(IntervalError):
```

```
    job_instance.thursday
with self.assertRaises(IntervalError):
    job_instance.friday
with self.assertRaises(IntervalError):
    job_instance.saturday
with self.assertRaises(IntervalError):
    job_instance.sunday

# test an invalid unit
job_instance.unit = "foo"
self.assertRaises(ScheduleValueError, job_instance.at, "1:0:0")
self.assertRaises(ScheduleValueError, job_instance._schedule_next_run)

# test start day exists but unit is not 'weeks'
job_instance.unit = "days"
job_instance.start_day = 1
self.assertRaises(ScheduleValueError, job_instance._schedule_next_run)

# test weeks with an invalid start day
job_instance.unit = "weeks"
job_instance.start_day = "bar"
self.assertRaises(ScheduleValueError, job_instance._schedule_next_run)

# test a valid unit with invalid hours/minutes/seconds
job_instance.unit = "days"
self.assertRaises(ScheduleValueError, job_instance.at, "25:00:00")
```

```
        self.assertRaises(ScheduleValueError, job_instance.at, "00:61:00")
        self.assertRaises(ScheduleValueError, job_instance.at, "00:00:61")

        # test invalid time format
        self.assertRaises(ScheduleValueError, job_instance.at, "25:0:0")
        self.assertRaises(ScheduleValueError, job_instance.at, "0:61:0")
        self.assertRaises(ScheduleValueError, job_instance.at, "0:0:61")

        # test (very specific) seconds with unspecified start_day
        job_instance.unit = "seconds"
        job_instance.at_time = datetime.datetime.now()
        job_instance.start_day = None
        self.assertRaises(ScheduleValueError, job_instance._schedule_next_run)

        # test self.latest >= self.interval
        job_instance.latest = 1
        self.assertRaises(ScheduleError, job_instance._schedule_next_run)
        job_instance.latest = 3
        self.assertRaises(ScheduleError, job_instance._schedule_next_run)

    def test_singular_time_units_match_plural_units(self):
        assert every().second.unit == every().seconds.unit
        assert every().minute.unit == every().minutes.unit
        assert every().hour.unit == every().hours.unit
        assert every().day.unit == every().days.unit
        assert every().week.unit == every().weeks.unit
```

```
def test_time_range(self):
    with mock_datetime(2014, 6, 28, 12, 0):
        mock_job = make_mock_job()

        # Choose a sample size large enough that it's unlikely the
        # same value will be chosen each time.
        minutes = set([
            every(5).to(30).minutes.do(mock_job).next_run.minute
            for i in range(100)
        ])

        assert len(minutes) > 1
        assert min(minutes) >= 5
        assert max(minutes) <= 30

def test_time_range_repr(self):
    mock_job = make_mock_job()

    with mock_datetime(2014, 6, 28, 12, 0):
        job_repr = repr(every(5).to(30).minutes.do(mock_job))

    assert job_repr.startswith('Every 5 to 30 minutes do job()')

def test_at_time(self):
    mock_job = make_mock_job()
```

```
assert every().day.at('10:30').do(mock_job).next_run.hour == 10

assert every().day.at('10:30').do(mock_job).next_run.minute == 30

assert every().day.at('10:30:50').do(mock_job).next_run.second == 50


self.assertRaises(ScheduleValueError, every().day.at, '2:30:000001')

self.assertRaises(ScheduleValueError, every().day.at, '::2')

self.assertRaises(ScheduleValueError, every().day.at, '.2')

self.assertRaises(ScheduleValueError, every().day.at, '2')

self.assertRaises(ScheduleValueError, every().day.at, ':2')

self.assertRaises(ScheduleValueError, every().day.at, ' 2:30:00')

self.assertRaises(ScheduleValueError, every().do, lambda: 0)

self.assertRaises(TypeError, every().day.at, 2)


# without a context manager, it incorrectly raises an error because

# it is not callable

with self.assertRaises(IntervalError):

    every(interval=2).second

with self.assertRaises(IntervalError):

    every(interval=2).minute

with self.assertRaises(IntervalError):

    every(interval=2).hour

with self.assertRaises(IntervalError):

    every(interval=2).day

with self.assertRaises(IntervalError):

    every(interval=2).week

with self.assertRaises(IntervalError):
```

```
            every(interval=2).monday
        with self.assertRaises(IntervalError):
            every(interval=2).tuesday
        with self.assertRaises(IntervalError):
            every(interval=2).wednesday
        with self.assertRaises(IntervalError):
            every(interval=2).thursday
        with self.assertRaises(IntervalError):
            every(interval=2).friday
        with self.assertRaises(IntervalError):
            every(interval=2).saturday
        with self.assertRaises(IntervalError):
            every(interval=2).sunday


    def test_at_time_hour(self):
        with mock_datetime(2010, 1, 6, 12, 20):
            mock_job = make_mock_job()
            assert every().hour.at(':30').do(mock_job).next_run.hour == 12
            assert every().hour.at(':30').do(mock_job).next_run.minute == 30
            assert every().hour.at(':30').do(mock_job).next_run.second == 0
            assert every().hour.at(':10').do(mock_job).next_run.hour == 13
            assert every().hour.at(':10').do(mock_job).next_run.minute == 10
            assert every().hour.at(':10').do(mock_job).next_run.second == 0
            assert every().hour.at(':00').do(mock_job).next_run.hour == 13
            assert every().hour.at(':00').do(mock_job).next_run.minute == 0
            assert every().hour.at(':00').do(mock_job).next_run.second == 0
```

```
        self.assertRaises(ScheduleValueError, every().hour.at, '2:30:00')
        self.assertRaises(ScheduleValueError, every().hour.at, '::2')
        self.assertRaises(ScheduleValueError, every().hour.at, '.2')
        self.assertRaises(ScheduleValueError, every().hour.at, '2')
        self.assertRaises(ScheduleValueError, every().hour.at, ' 2:30')
        self.assertRaises(ScheduleValueError, every().hour.at, "61:00")
        self.assertRaises(ScheduleValueError, every().hour.at, "00:61")
        self.assertRaises(ScheduleValueError, every().hour.at, "01:61")
        self.assertRaises(TypeError, every().hour.at, 2)

def test_at_time_minute(self):
    with mock_datetime(2010, 1, 6, 12, 20, 30):
        mock_job = make_mock_job()
        assert every().minute.at(':40').do(mock_job).next_run.hour == 12
        assert every().minute.at(':40').do(mock_job).next_run.minute == 20
        assert every().minute.at(':40').do(mock_job).next_run.second == 40
        assert every().minute.at(':10').do(mock_job).next_run.hour == 12
        assert every().minute.at(':10').do(mock_job).next_run.minute == 21
        assert every().minute.at(':10').do(mock_job).next_run.second == 10

        self.assertRaises(ScheduleValueError, every().minute.at, '::2')
        self.assertRaises(ScheduleValueError, every().minute.at, '.2')
        self.assertRaises(ScheduleValueError, every().minute.at, '2')
        self.assertRaises(ScheduleValueError, every().minute.at, '2:30:00')
        self.assertRaises(ScheduleValueError, every().minute.at, '2:30')
```

```
        self.assertRaises(ScheduleValueError, every().minute.at, ' :30')
        self.assertRaises(TypeError, every().minute.at, 2)


def test_next_run_time(self):
    with mock_datetime(2010, 1, 6, 12, 15):
        mock_job = make_mock_job()
        assert schedule.next_run() is None
        assert every().minute.do(mock_job).next_run.minute == 16
        assert every(5).minutes.do(mock_job).next_run.minute == 20
        assert every().hour.do(mock_job).next_run.hour == 13
        assert every().day.do(mock_job).next_run.day == 7
        assert every().day.at('09:00').do(mock_job).next_run.day == 7
        assert every().day.at('12:30').do(mock_job).next_run.day == 6
        assert every().week.do(mock_job).next_run.day == 13
        assert every().monday.do(mock_job).next_run.day == 11
        assert every().tuesday.do(mock_job).next_run.day == 12
        assert every().wednesday.do(mock_job).next_run.day == 13
        assert every().thursday.do(mock_job).next_run.day == 7
        assert every().friday.do(mock_job).next_run.day == 8
        assert every().saturday.do(mock_job).next_run.day == 9
        assert every().sunday.do(mock_job).next_run.day == 10


def test_run_all(self):
    mock_job = make_mock_job()
    every().minute.do(mock_job)
    every().hour.do(mock_job)
```

```
    every().day.at('11:00').do(mock_job)
    schedule.run_all()
    assert mock_job.call_count == 3

def test_job_func_args_are_passed_on(self):
    mock_job = make_mock_job()
    every().second.do(mock_job, 1, 2, 'three', foo=23, bar={})
    schedule.run_all()
    mock_job.assert_called_once_with(1, 2, 'three', foo=23, bar={})

def test_to_string(self):
    def job_fun():
        pass
    s = str(every().minute.do(job_fun, 'foo', bar=23))
    assert s == ("Job(interval=1, unit=minutes, do=job_fun, "
                 "args=('foo',), kwargs={'bar': 23})")
    assert 'job_fun' in s
    assert 'foo' in s
    assert '{'bar': 23}' in s

def test_to_repr(self):
    def job_fun():
        pass
    s = repr(every().minute.do(job_fun, 'foo', bar=23))
    assert s.startswith("Every 1 minute do job_fun('foo', bar=23) "
                        "(last run: [never], next run: ")
```

```
    assert 'job_fun' in s
    assert 'foo' in s
    assert 'bar=23' in s

    # test repr when at_time is not None
    s2 = repr(every().day.at("00:00").do(job_fun, 'foo', bar=23))
    assert s2.startswith(("Every 1 day at 00:00:00 do job_fun('foo', "
                          "bar=23) (last run: [never], next run: "))

def test_to_string_lambda_job_func(self):
    assert len(str(every().minute.do(lambda: 1))) > 1
    assert len(str(every().day.at('10:30').do(lambda: 1))) > 1

def test_to_string_functools_partial_job_func(self):
    def job_fun(arg):
        pass
    job_fun = functools.partial(job_fun, 'foo')
    job_repr = repr(every().minute.do(job_fun, bar=True, somekey=23))
    assert 'functools.partial' in job_repr
    assert 'bar=True' in job_repr
    assert 'somekey=23' in job_repr

def test_run_pending(self):
    """Check that run_pending() runs pending jobs.
    We do this by overriding datetime.datetime with mock objects
    that represent increasing system times.
```

```
Please note that it is *intended behavior that run_pending() does not
run missed jobs*. For example, if you've registered a job that
should run every minute and you only call run_pending() in one hour
increments then your job won't be run 60 times in between but
only once.
"""
mock_job = make_mock_job()

with mock_datetime(2010, 1, 6, 12, 15):
    every().minute.do(mock_job)
    every().hour.do(mock_job)
    every().day.do(mock_job)
    every().sunday.do(mock_job)
    schedule.run_pending()
    assert mock_job.call_count == 0

with mock_datetime(2010, 1, 6, 12, 16):
    schedule.run_pending()
    assert mock_job.call_count == 1

with mock_datetime(2010, 1, 6, 13, 16):
    mock_job.reset_mock()
    schedule.run_pending()
    assert mock_job.call_count == 2
```

```
    with mock_datetime(2010, 1, 7, 13, 16):
        mock_job.reset_mock()
        schedule.run_pending()
        assert mock_job.call_count == 3

    with mock_datetime(2010, 1, 10, 13, 16):
        mock_job.reset_mock()
        schedule.run_pending()
        assert mock_job.call_count == 4

def test_run_every_weekday_at_specific_time_today(self):
    mock_job = make_mock_job()
    with mock_datetime(2010, 1, 6, 13, 16):
        every().wednesday.at('14:12').do(mock_job)
        schedule.run_pending()
        assert mock_job.call_count == 0

    with mock_datetime(2010, 1, 6, 14, 16):
        schedule.run_pending()
        assert mock_job.call_count == 1

def test_run_every_weekday_at_specific_time_past_today(self):
    mock_job = make_mock_job()
    with mock_datetime(2010, 1, 6, 13, 16):
        every().wednesday.at('13:15').do(mock_job)
        schedule.run_pending()
```

```
            assert mock_job.call_count == 0

        with mock_datetime(2010, 1, 13, 13, 14):
            schedule.run_pending()
            assert mock_job.call_count == 0

        with mock_datetime(2010, 1, 13, 13, 16):
            schedule.run_pending()
            assert mock_job.call_count == 1

    def test_run_every_n_days_at_specific_time(self):
        mock_job = make_mock_job()
        with mock_datetime(2010, 1, 6, 11, 29):
            every(2).days.at('11:30').do(mock_job)
            schedule.run_pending()
            assert mock_job.call_count == 0

        with mock_datetime(2010, 1, 6, 11, 31):
            schedule.run_pending()
            assert mock_job.call_count == 0

        with mock_datetime(2010, 1, 7, 11, 31):
            schedule.run_pending()
            assert mock_job.call_count == 0

        with mock_datetime(2010, 1, 8, 11, 29):
```

```
            schedule.run_pending()
            assert mock_job.call_count == 0

        with mock_datetime(2010, 1, 8, 11, 31):
            schedule.run_pending()
            assert mock_job.call_count == 1

        with mock_datetime(2010, 1, 10, 11, 31):
            schedule.run_pending()
            assert mock_job.call_count == 2

    def test_next_run_property(self):
        original_datetime = datetime.datetime
        with mock_datetime(2010, 1, 6, 13, 16):
            hourly_job = make_mock_job('hourly')
            daily_job = make_mock_job('daily')
            every().day.do(daily_job)
            every().hour.do(hourly_job)
            assert len(schedule.jobs) == 2
            # Make sure the hourly job is first
            assert schedule.next_run() == original_datetime(2010, 1, 6, 14, 16)
            assert schedule.idle_seconds() == 60 * 60

    def test_cancel_job(self):
        def stop_job():
            return schedule.CancelJob
```

```
    mock_job = make_mock_job()

    every().second.do(stop_job)
    mj = every().second.do(mock_job)
    assert len(schedule.jobs) == 2

    schedule.run_all()
    assert len(schedule.jobs) == 1
    assert schedule.jobs[0] == mj

    schedule.cancel_job('Not a job')
    assert len(schedule.jobs) == 1
    schedule.default_scheduler.cancel_job('Not a job')
    assert len(schedule.jobs) == 1

    schedule.cancel_job(mj)
    assert len(schedule.jobs) == 0

def test_cancel_jobs(self):
    def stop_job():
        return schedule.CancelJob

    every().second.do(stop_job)
    every().second.do(stop_job)
    every().second.do(stop_job)
    assert len(schedule.jobs) == 3
```

```
    schedule.run_all()
    assert len(schedule.jobs) == 0

def test_tag_type_enforcement(self):
    job1 = every().second.do(make_mock_job(name='job1'))
    self.assertRaises(TypeError, job1.tag, {})
    self.assertRaises(TypeError, job1.tag, 1, 'a', [])
    job1.tag(0, 'a', True)
    assert len(job1.tags) == 3

def test_clear_by_tag(self):
    every().second.do(make_mock_job(name='job1')).tag('tag1')
    every().second.do(make_mock_job(name='job2')).tag('tag1', 'tag2')
    every().second.do(make_mock_job(name='job3')).tag('tag3', 'tag3',
                                          'tag3', 'tag2')
    assert len(schedule.jobs) == 3
    schedule.run_all()
    assert len(schedule.jobs) == 3
    schedule.clear('tag3')
    assert len(schedule.jobs) == 2
    schedule.clear('tag1')
    assert len(schedule.jobs) == 0
    every().second.do(make_mock_job(name='job1'))
    every().second.do(make_mock_job(name='job2'))
    every().second.do(make_mock_job(name='job3'))
```

```
    schedule.clear()

    assert len(schedule.jobs) == 0


def test_misconfigured_job_wont_break_scheduler(self):
    """
    Ensure an interrupted job definition chain won't break
    the scheduler instance permanently.
    """
    scheduler = schedule.Scheduler()
    scheduler.every()
    scheduler.every(10).seconds
    scheduler.run_pending()
```

```
[tox]

envlist = py27, py3{5,6,7,8}, docs

skip_missing_interpreters = true



[tox:travis]

2.7 = py27, docs

3.5 = py35, docs

3.6 = py36, docs

3.7 = py37, docs

3.8 = py38, docs



[testenv]

deps = -rrequirements-dev.txt

commands =

    py.test test_schedule.py --flake8 schedule -v --cov schedule --cov-report term-missing

    python setup.py check --strict --metadata --restructuredtext



[testenv:docs]

changedir = docs

deps = -rrequirements-dev.txt

commands =

    sphinx-build -W -b html -d {envtmpdir}/doctrees . {envtmpdir}/html
```

.. _api:

Developer Interface

==========

.. module:: schedule

This part of the documentation covers all the interfaces of schedule. For

parts where schedule depends on external libraries, we document the most

important right here and provide links to the canonical documentation.

Main Interface

.. autodata:: default_scheduler

.. autodata:: jobs

.. autofunction:: every

.. autofunction:: run_pending

.. autofunction:: run_all

.. autofunction:: clear

.. autofunction:: cancel_job

.. autofunction:: next_run

.. autofunction:: idle_seconds

Exceptions

.. autoexception:: schedule.CancelJob

Classes

.. autoclass:: schedule.Scheduler

   :members:

   :undoc-members:

.. autoclass:: schedule.Job

   :members:

   :undoc-members:

```
# -*- coding: utf-8 -*-
#
# schedule documentation build configuration file, created by
# sphinx-quickstart on Mon Nov  7 15:14:48 2016.
#
# This file is execfile()d with the current directory set to its
# containing dir.
#
# Note that not all possible configuration values are present in this
# autogenerated file.
#
# All configuration values have a default; values that are commented out
# serve to show the default.


# If extensions (or modules to document with autodoc) are in another directory,
# add these directories to sys.path here. If the directory is relative to the
# documentation root, use os.path.abspath to make it absolute, like shown here.
#
# (schedule modules lives up one level from docs/)
#
import os
import sys
sys.path.insert(0, os.path.abspath('..'))


# -- General configuration ------------------------------------------------
```

```
# If your documentation needs a minimal Sphinx version, state it here.
#
# needs_sphinx = '1.0'

# Add any Sphinx extension module names here, as strings. They can be
# extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
# ones.
extensions = [
    'sphinx.ext.autodoc',
    'sphinx.ext.todo',
    'sphinx.ext.coverage',
    'sphinx.ext.viewcode',
    # 'sphinx.ext.githubpages',  # This breaks the ReadTheDocs build
]

# Add any paths that contain templates here, relative to this directory.
templates_path = ['_templates']

# The suffix(es) of source filenames.
# You can specify multiple suffix as a list of string:
#
# source_suffix = ['.rst', '.md']
source_suffix = '.rst'

# The encoding of source files.
#
```

```
# source_encoding = 'utf-8-sig'

# The master toctree document.
master_doc = 'index'

# General information about the project.
project = u'schedule'
copyright = u'2016, <a href="https://dbader.org/">Daniel Bader</a>'
author = u'<a href="https://dbader.org/">Daniel Bader</a>'

# The version info for the project you're documenting, acts as replacement for
# |version| and |release|, also used in various other places throughout the
# built documents.
#
# The short X.Y version.
version = u'0.4.0'
# The full version, including alpha/beta/rc tags.
release = u'0.4.0'

# The language for content autogenerated by Sphinx. Refer to documentation
# for a list of supported languages.
#
# This is also used if you do content translation via gettext catalogs.
# Usually you set "language" from the command line for these cases.
language = None
```

```
# There are two options for replacing |today|: either, you set today to some
# non-false value, then it is used:
#
# today = ''
#
# Else, today_fmt is used as the format for a strftime call.
#
# today_fmt = '%B %d, %Y'

# List of patterns, relative to source directory, that match files and
# directories to ignore when looking for source files.
# This patterns also effect to html_static_path and html_extra_path
exclude_patterns = ['_build', 'Thumbs.db', '.DS_Store']

# The reST default role (used for this markup: `text`) to use for all
# documents.
#
# default_role = None

# If true, '()' will be appended to :func: etc. cross-reference text.
#
# add_function_parentheses = True

# If true, the current module name will be prepended to all description
# unit titles (such as .. function::).
#
```

```
# add_module_names = True

# If true, sectionauthor and moduleauthor directives will be shown in the
# output. They are ignored by default.
#
# show_authors = False

# The name of the Pygments (syntax highlighting) style to use.
# pygments_style = 'flask_theme_support.FlaskyStyle'

# A list of ignored prefixes for module index sorting.
# modindex_common_prefix = []

# If true, keep warnings as "system message" paragraphs in the built documents.
# keep_warnings = False

# If true, `todo` and `todoList` produce output, else they produce nothing.
todo_include_todos = True


# -- Options for HTML output ----------------------------------------------

# The theme to use for HTML and HTML Help pages.  See the documentation for
# a list of builtin themes.
#
html_theme = 'alabaster'
```

```
# Theme options are theme-specific and customize the look and feel of a theme

# further.  For a list of options available for each theme, see the

# documentation.

#

html_theme_options = {

    'show_powered_by': False,

    'github_user': 'dbader',

    'github_repo': 'schedule',

    'github_banner': True,

    'show_related': False

}


# Add any paths that contain custom themes here, relative to this directory.

# html_theme_path = []


# The name for this set of Sphinx documents.

# "<project> v<release> documentation" by default.

#

# html_title = u'schedule v0.4.0'


# A shorter title for the navigation bar.  Default is the same as html_title.

#

# html_short_title = None


# The name of an image file (relative to this directory) to place at the top
```

```
# of the sidebar.
#
# html_logo = None


# The name of an image file (relative to this directory) to use as a favicon of
# the docs.  This file should be a Windows icon file (.ico) being 16x16 or 32x32
# pixels large.
#
# html_favicon = None


# Add any paths that contain custom static files (such as style sheets) here,
# relative to this directory. They are copied after the builtin static files,
# so a file named "default.css" will overwrite the builtin "default.css".
html_static_path = ['_static']


# Add any extra paths that contain custom files (such as robots.txt or
# .htaccess) here, relative to this directory. These files are copied
# directly to the root of the documentation.
#
# html_extra_path = []


# If not None, a 'Last updated on:' timestamp is inserted at every page
# bottom, using the given strftime format.
# The empty string is equivalent to '%b %d, %Y'.
#
# html_last_updated_fmt = None
```

```
# If true, SmartyPants will be used to convert quotes and dashes to

# typographically correct entities.

#

html_use_smartypants = True


# Custom sidebar templates, maps document names to template names.

#

html_sidebars = {

    'index':    ['sidebarintro.html', 'sourcelink.html', 'searchbox.html'],

}


# Additional templates that should be rendered to pages, maps page names to

# template names.

#

# html_additional_pages = {}


# If false, no module index is generated.

#

# html_domain_indices = True


# If false, no index is generated.

#

# html_use_index = True


# If true, the index is split into individual pages for each letter.
```

```
#

# html_split_index = False


# If true, links to the reST sources are added to the pages.

#

html_show_sourcelink = False


# If true, "Created using Sphinx" is shown in the HTML footer. Default is True.

#

html_show_sphinx = False


# If true, "(C) Copyright ..." is shown in the HTML footer. Default is True.

#

html_show_copyright = True


# If true, an OpenSearch description file will be output, and all pages will

# contain a <link> tag referring to it.  The value of this option must be the

# base URL from which the finished HTML is served.

#

# html_use_opensearch = ''


# This is the file name suffix for HTML files (e.g. ".xhtml").

# html_file_suffix = None


# Language to be used for generating the HTML full-text search index.

# Sphinx supports the following languages:
```

```
#   'da', 'de', 'en', 'es', 'fi', 'fr', 'hu', 'it', 'ja'
#   'nl', 'no', 'pt', 'ro', 'ru', 'sv', 'tr', 'zh'
#
# html_search_language = 'en'

# A dictionary with options for the search language support, empty by default.
# 'ja' uses this config value.
# 'zh' user can custom change `jieba` dictionary path.
#
# html_search_options = {'type': 'default'}

# The name of a javascript file (relative to the configuration directory) that
# implements a search results scorer. If empty, the default will be used.
#
# html_search_scorer = 'scorer.js'

# Output file base name for HTML help builder.
htmlhelp_basename = 'scheduledoc'

# -- Options for LaTeX output ---------------------------------------------

latex_elements = {
    # The paper size ('letterpaper' or 'a4paper').
    #
    # 'papersize': 'letterpaper',
```

```
    # The font size ('10pt', '11pt' or '12pt').
    #
    # 'pointsize': '10pt',

    # Additional stuff for the LaTeX preamble.
    #
    # 'preamble': '',

    # Latex figure (float) alignment
    #
    # 'figure_align': 'htbp',
}

# Grouping the document tree into LaTeX files. List of tuples
# (source start file, target name, title,
#  author, documentclass [howto, manual, or own class]).
latex_documents = [
   (master_doc, 'schedule.tex', u'schedule Documentation',
    u'Daniel Bader', 'manual'),
]

# The name of an image file (relative to this directory) to place at the top of
# the title page.
#
# latex_logo = None
```

```
# For "manual" documents, if this is true, then toplevel headings are parts,
# not chapters.
#
# latex_use_parts = False


# If true, show page references after internal links.
#
# latex_show_pagerefs = False


# If true, show URL addresses after external links.
#
# latex_show_urls = False


# Documents to append as an appendix to all manuals.
#
# latex_appendices = []


# It false, will not define \strong, \code,  itleref, \crossref ... but only
# \sphinxstrong, ..., \sphinxtitleref, ... To help avoid clash with user added
# packages.
#
# latex_keep_old_macro_names = True


# If false, no module index is generated.
#
# latex_domain_indices = True
```

```
# -- Options for manual page output ---------------------------------------


# One entry per manual page. List of tuples

# (source start file, name, description, authors, manual section).

man_pages = [

    (master_doc, 'schedule', u'schedule Documentation',

     [author], 1)

]


# If true, show URL addresses after external links.

#

# man_show_urls = False




# -- Options for Texinfo output -------------------------------------------


# Grouping the document tree into Texinfo files. List of tuples

# (source start file, target name, title, author,

#  dir menu entry, description, category)

texinfo_documents = [

    (master_doc, 'schedule', u'schedule Documentation',

     author, 'schedule', 'One line description of project.',

     'Miscellaneous'),

]
```

```
# Documents to append as an appendix to all manuals.
#
# texinfo_appendices = []

# If false, no module index is generated.
#
# texinfo_domain_indices = True

# How to display URL addresses: 'footnote', 'no', or 'inline'.
#
# texinfo_show_urls = 'footnote'

# If true, do not generate a @detailmenu in the "Top" node's menu.
#
# texinfo_no_detailmenu = False

autodoc_member_order = 'bysource'

# We're pulling in some external images like CI badges.
suppress_warnings = ['image.nonlocal_uri']
```

.. At some point we'll want to migrate FAQ.rst to the docs folder but to

.. breaking links on PyPI we need to leave it there until we prepare the

.. next schedule release

.. include:: ../FAQ.rst

schedule

========

.. image:: https://api.travis-ci.org/dbader/schedule.svg?branch=master

:target: https://travis-ci.org/dbader/schedule

.. image:: https://coveralls.io/repos/dbader/schedule/badge.svg?branch=master

:target: https://coveralls.io/r/dbader/schedule

.. image:: https://img.shields.io/pypi/v/schedule.svg

:target: https://pypi.python.org/pypi/schedule

Python job scheduling for humans.

An in-process scheduler for periodic jobs that uses the builder pattern

for configuration. Schedule lets you run Python functions (or any other

callable) periodically at pre-determined intervals using a simple,

human-friendly syntax.

Inspired by `Adam Wiggins' <https://github.com/adamwiggins>`_ article `"Rethinking Cron" <https://adam.herokuapp.com/past/2

Features

- A simple to use API for scheduling jobs.

- Very lightweight and no external dependencies.

- Excellent test coverage.

- Tested on Python 2.7, 3.5, and 3.6

Usage

.. code-block:: bash

    $ pip install schedule

.. code-block:: python

    import schedule
    import time

    def job():
        print("I'm working...")

    schedule.every(10).minutes.do(job)
    schedule.every().hour.do(job)
    schedule.every().day.at("10:30").do(job)
    schedule.every().monday.do(job)
    schedule.every().wednesday.at("13:15").do(job)
    schedule.every().minute.at(":17").do(job)

    while True:

```
    schedule.run_pending()
    time.sleep(1)
```

API Documentation

If you are looking for information on a specific function, class, or method,

this part of the documentation is for you.

.. toctree::

   api

Common Questions

Please check here before creating a new issue ticket.

.. toctree::

   faq

Issues

If you encounter any problems, please `file an issue <http://github.com/dbader/schedule/issues>`_ along with a detailed descript

About Schedule

Schedule was created by `Daniel Bader <https://dbader.org/>`__ - `@dbader_org <https://twitter.com/dbader_org>`_

.. include:: ../AUTHORS.rst

```
# Makefile for Sphinx documentation
#

# You can set these variables from the command line.
SPHINXOPTS    =
SPHINXBUILD   = sphinx-build
PAPER         =
BUILDDIR      = _build

# Internal variables.
PAPEROPT_a4     = -D latex_paper_size=a4
PAPEROPT_letter = -D latex_paper_size=letter
ALLSPHINXOPTS   = -d $(BUILDDIR)/doctrees $(PAPEROPT_$(PAPER)) $(SPHINXOPTS) .
# the i18n builder cannot share the environment and doctrees with the others
I18NSPHINXOPTS  = $(PAPEROPT_$(PAPER)) $(SPHINXOPTS) .

.PHONY: help
help:
 @echo "Please use \`make <target>' where <target> is one of"
 @echo "  html       to make standalone HTML files"
 @echo "  dirhtml    to make HTML files named index.html in directories"
 @echo "  singlehtml to make a single large HTML file"
 @echo "  pickle     to make pickle files"
 @echo "  json       to make JSON files"
 @echo "  htmlhelp   to make HTML files and a HTML help project"
 @echo "  qthelp     to make HTML files and a qthelp project"
```

```
	@echo "  applehelp  to make an Apple Help Book"
	@echo "  devhelp    to make HTML files and a Devhelp project"
	@echo "  epub       to make an epub"
	@echo "  epub3      to make an epub3"
	@echo "  latex      to make LaTeX files, you can set PAPER=a4 or PAPER=letter"
	@echo "  latexpdf   to make LaTeX files and run them through pdflatex"
	@echo "  latexpdfja to make LaTeX files and run them through platex/dvipdfmx"
	@echo "  text       to make text files"
	@echo "  man        to make manual pages"
	@echo "  texinfo    to make Texinfo files"
	@echo "  info       to make Texinfo files and run them through makeinfo"
	@echo "  gettext    to make PO message catalogs"
	@echo "  changes    to make an overview of all changed/added/deprecated items"
	@echo "  xml        to make Docutils-native XML files"
	@echo "  pseudoxml  to make pseudoxml-XML files for display purposes"
	@echo "  linkcheck  to check all external links for integrity"
	@echo "  doctest    to run all doctests embedded in the documentation (if enabled)"
	@echo "  coverage   to run coverage check of the documentation (if enabled)"
	@echo "  dummy      to check syntax errors of document sources"

.PHONY: clean
clean:
	rm -rf $(BUILDDIR)/*

.PHONY: html
html:
```

```
	$(SPHINXBUILD) -b html $(ALLSPHINXOPTS) $(BUILDDIR)/html
	@echo
	@echo "Build finished. The HTML pages are in $(BUILDDIR)/html."

.PHONY: dirhtml
dirhtml:
	$(SPHINXBUILD) -b dirhtml $(ALLSPHINXOPTS) $(BUILDDIR)/dirhtml
	@echo
	@echo "Build finished. The HTML pages are in $(BUILDDIR)/dirhtml."

.PHONY: singlehtml
singlehtml:
	$(SPHINXBUILD) -b singlehtml $(ALLSPHINXOPTS) $(BUILDDIR)/singlehtml
	@echo
	@echo "Build finished. The HTML page is in $(BUILDDIR)/singlehtml."

.PHONY: pickle
pickle:
	$(SPHINXBUILD) -b pickle $(ALLSPHINXOPTS) $(BUILDDIR)/pickle
	@echo
	@echo "Build finished; now you can process the pickle files."

.PHONY: json
json:
	$(SPHINXBUILD) -b json $(ALLSPHINXOPTS) $(BUILDDIR)/json
	@echo
```

```
	@echo "Build finished; now you can process the JSON files."

.PHONY: htmlhelp
htmlhelp:
	$(SPHINXBUILD) -b htmlhelp $(ALLSPHINXOPTS) $(BUILDDIR)/htmlhelp
	@echo
	@echo "Build finished; now you can run HTML Help Workshop with the" \
	      ".hhp project file in $(BUILDDIR)/htmlhelp."

.PHONY: qthelp
qthelp:
	$(SPHINXBUILD) -b qthelp $(ALLSPHINXOPTS) $(BUILDDIR)/qthelp
	@echo
	@echo "Build finished; now you can run "qcollectiongenerator" with the" \
	      ".qhcp project file in $(BUILDDIR)/qthelp, like this:"
	@echo "# qcollectiongenerator $(BUILDDIR)/qthelp/schedule.qhcp"
	@echo "To view the help file:"
	@echo "# assistant -collectionFile $(BUILDDIR)/qthelp/schedule.qhc"

.PHONY: applehelp
applehelp:
	$(SPHINXBUILD) -b applehelp $(ALLSPHINXOPTS) $(BUILDDIR)/applehelp
	@echo
	@echo "Build finished. The help book is in $(BUILDDIR)/applehelp."
	@echo "N.B. You won't be able to view it unless you put it in" \
	      "~/Library/Documentation/Help or install it in your application" \
```

```
	"bundle."


.PHONY: devhelp

devhelp:

 $(SPHINXBUILD) -b devhelp $(ALLSPHINXOPTS) $(BUILDDIR)/devhelp

 @echo

 @echo "Build finished."

 @echo "To view the help file:"

 @echo "# mkdir -p $$HOME/.local/share/devhelp/schedule"

 @echo "# ln -s $(BUILDDIR)/devhelp $$HOME/.local/share/devhelp/schedule"

 @echo "# devhelp"


.PHONY: epub

epub:

 $(SPHINXBUILD) -b epub $(ALLSPHINXOPTS) $(BUILDDIR)/epub

 @echo

 @echo "Build finished. The epub file is in $(BUILDDIR)/epub."


.PHONY: epub3

epub3:

 $(SPHINXBUILD) -b epub3 $(ALLSPHINXOPTS) $(BUILDDIR)/epub3

 @echo

 @echo "Build finished. The epub3 file is in $(BUILDDIR)/epub3."


.PHONY: latex

latex:
```

```
	$(SPHINXBUILD) -b latex $(ALLSPHINXOPTS) $(BUILDDIR)/latex
	@echo
	@echo "Build finished; the LaTeX files are in $(BUILDDIR)/latex."
	@echo "Run \`make' in that directory to run these through (pdf)latex" \
	      "(use \`make latexpdf' here to do that automatically)."

.PHONY: latexpdf
latexpdf:
	$(SPHINXBUILD) -b latex $(ALLSPHINXOPTS) $(BUILDDIR)/latex
	@echo "Running LaTeX files through pdflatex..."
	$(MAKE) -C $(BUILDDIR)/latex all-pdf
	@echo "pdflatex finished; the PDF files are in $(BUILDDIR)/latex."

.PHONY: latexpdfja
latexpdfja:
	$(SPHINXBUILD) -b latex $(ALLSPHINXOPTS) $(BUILDDIR)/latex
	@echo "Running LaTeX files through platex and dvipdfmx..."
	$(MAKE) -C $(BUILDDIR)/latex all-pdf-ja
	@echo "pdflatex finished; the PDF files are in $(BUILDDIR)/latex."

.PHONY: text
text:
	$(SPHINXBUILD) -b text $(ALLSPHINXOPTS) $(BUILDDIR)/text
	@echo
	@echo "Build finished. The text files are in $(BUILDDIR)/text."
```

```
.PHONY: man

man:
	$(SPHINXBUILD) -b man $(ALLSPHINXOPTS) $(BUILDDIR)/man
	@echo
	@echo "Build finished. The manual pages are in $(BUILDDIR)/man."


.PHONY: texinfo

texinfo:
	$(SPHINXBUILD) -b texinfo $(ALLSPHINXOPTS) $(BUILDDIR)/texinfo
	@echo
	@echo "Build finished. The Texinfo files are in $(BUILDDIR)/texinfo."
	@echo "Run \`make' in that directory to run these through makeinfo" \
	      "(use \`make info' here to do that automatically)."


.PHONY: info

info:
	$(SPHINXBUILD) -b texinfo $(ALLSPHINXOPTS) $(BUILDDIR)/texinfo
	@echo "Running Texinfo files through makeinfo..."
	make -C $(BUILDDIR)/texinfo info
	@echo "makeinfo finished; the Info files are in $(BUILDDIR)/texinfo."


.PHONY: gettext

gettext:
	$(SPHINXBUILD) -b gettext $(I18NSPHINXOPTS) $(BUILDDIR)/locale
	@echo
	@echo "Build finished. The message catalogs are in $(BUILDDIR)/locale."
```

```
.PHONY: changes

changes:

 $(SPHINXBUILD) -b changes $(ALLSPHINXOPTS) $(BUILDDIR)/changes

 @echo

 @echo "The overview file is in $(BUILDDIR)/changes."


.PHONY: linkcheck

linkcheck:

 $(SPHINXBUILD) -b linkcheck $(ALLSPHINXOPTS) $(BUILDDIR)/linkcheck

 @echo

 @echo "Link check complete; look for any errors in the above output " \

       "or in $(BUILDDIR)/linkcheck/output.txt."


.PHONY: doctest

doctest:

 $(SPHINXBUILD) -b doctest $(ALLSPHINXOPTS) $(BUILDDIR)/doctest

 @echo "Testing of doctests in the sources finished, look at the " \

       "results in $(BUILDDIR)/doctest/output.txt."


.PHONY: coverage

coverage:

 $(SPHINXBUILD) -b coverage $(ALLSPHINXOPTS) $(BUILDDIR)/coverage

 @echo "Testing of coverage in the sources finished, look at the " \

       "results in $(BUILDDIR)/coverage/python.txt."
```

```
.PHONY: xml

xml:

 $(SPHINXBUILD) -b xml $(ALLSPHINXOPTS) $(BUILDDIR)/xml

 @echo

 @echo "Build finished. The XML files are in $(BUILDDIR)/xml."


.PHONY: pseudoxml

pseudoxml:

 $(SPHINXBUILD) -b pseudoxml $(ALLSPHINXOPTS) $(BUILDDIR)/pseudoxml

 @echo

 @echo "Build finished. The pseudo-XML files are in $(BUILDDIR)/pseudoxml."


.PHONY: dummy

dummy:

 $(SPHINXBUILD) -b dummy $(ALLSPHINXOPTS) $(BUILDDIR)/dummy

 @echo

 @echo "Build finished. Dummy builder generates no files."
```

don't mind me

<iframe src="https://ghbtns.com/github-btn.html?user=dbader&repo=schedule&type=watch&count=true&size=large" allowtransp

<h3>ðŸ“° Useful Links</h3>

  Schedule @ GitHub

  Schedule @ PyPI

  Issue Tracker

<h3>ðŸ•• More Python</h3>

<p>Follow @dbader_org <sc

  Dan's Python Screencasts

  Dan's Python Newsletter

  Dan's Python Tutorials

"""

Python job scheduling for humans.

github.com/dbader/schedule

An in-process scheduler for periodic jobs that uses the builder pattern
for configuration. Schedule lets you run Python functions (or any other
callable) periodically at pre-determined intervals using a simple,
human-friendly syntax.

Inspired by Addam Wiggins' article "Rethinking Cron" [1] and the
"clockwork" Ruby module [2][3].

Features:

    - A simple to use API for scheduling jobs.

    - Very lightweight and no external dependencies.

    - Excellent test coverage.

    - Tested on Python 2.7, 3.5 and 3.6

Usage:

    >>> import schedule

    >>> import time

    >>> def job(message='stuff'):

    >>>   print("I'm working on:", message)

```
    >>> schedule.every(10).minutes.do(job)
    >>> schedule.every(5).to(10).days.do(job)
    >>> schedule.every().hour.do(job, message='things')
    >>> schedule.every().day.at("10:30").do(job)

    >>> while True:
    >>>     schedule.run_pending()
    >>>     time.sleep(1)

[1] https://adam.herokuapp.com/past/2010/4/13/rethinking_cron/
[2] https://github.com/Rykian/clockwork
[3] https://adam.herokuapp.com/past/2010/6/30/replace_cron_with_clockwork/
"""
try:
    from collections.abc import Hashable
except ImportError:
    from collections import Hashable
import datetime
import functools
import logging
import random
import re
import time

logger = logging.getLogger('schedule')
```

```
class ScheduleError(Exception):
    """Base schedule exception"""
    pass



class ScheduleValueError(ScheduleError):
    """Base schedule value error"""
    pass



class IntervalError(ScheduleValueError):
    """An improper interval was used"""
    pass



class CancelJob(object):
    """
    Can be returned from a job to unschedule itself.
    """
    pass



class Scheduler(object):
    """
    Objects instantiated by the :class:`Scheduler <Scheduler>` are
```

```
factories to create jobs, keep record of scheduled jobs and
handle their execution.
"""
def __init__(self):
    self.jobs = []

def run_pending(self):
    """
    Run all jobs that are scheduled to run.

    Please note that it is *intended behavior that run_pending()
    does not run missed jobs*. For example, if you've registered a job
    that should run every minute and you only call run_pending()
    in one hour increments then your job won't be run 60 times in
    between but only once.
    """
    runnable_jobs = (job for job in self.jobs if job.should_run)
    for job in sorted(runnable_jobs):
        self._run_job(job)

def run_all(self, delay_seconds=0):
    """
    Run all jobs regardless if they are scheduled to run or not.

    A delay of `delay` seconds is added between each job. This helps
    distribute system load generated by the jobs more evenly
```

```
    over time.

    :param delay_seconds: A delay added between every executed job
    """
    logger.info('Running *all* %i jobs with %is delay inbetween',
                len(self.jobs), delay_seconds)
    for job in self.jobs[:]:
        self._run_job(job)
        time.sleep(delay_seconds)

def clear(self, tag=None):
    """
    Deletes scheduled jobs marked with the given tag, or all jobs
    if tag is omitted.

    :param tag: An identifier used to identify a subset of
                jobs to delete
    """
    if tag is None:
        del self.jobs[:]
    else:
        self.jobs[:] = (job for job in self.jobs if tag not in job.tags)

def cancel_job(self, job):
    """
    Delete a scheduled job.
```

```
    :param job: The job to be unscheduled
    """
    try:
        self.jobs.remove(job)
    except ValueError:
        pass


def every(self, interval=1):
    """
    Schedule a new periodic job.

    :param interval: A quantity of a certain time unit
    :return: An unconfigured :class:`Job <Job>`
    """
    job = Job(interval, self)
    return job


def _run_job(self, job):
    ret = job.run()
    if isinstance(ret, CancelJob) or ret is CancelJob:
        self.cancel_job(job)


@property
def next_run(self):
    """
```

```
        Datetime when the next job should run.

        :return: A :class:`~datetime.datetime` object
        """
        if not self.jobs:
            return None
        return min(self.jobs).next_run

    @property
    def idle_seconds(self):
        """
        :return: Number of seconds until
                 :meth:`next_run <Scheduler.next_run>`.
        """
        return (self.next_run - datetime.datetime.now()).total_seconds()


class Job(object):
    """
    A periodic job as used by :class:`Scheduler`.

    :param interval: A quantity of a certain time unit
    :param scheduler: The :class:`Scheduler <Scheduler>` instance that
                      this job will register itself with once it has
                      been fully configured in :meth:`Job.do()`.
```

```
Every job runs at a given fixed time interval that is defined by:

* a :meth:`time unit <Job.second>`
* a quantity of `time units` defined by `interval`

A job is usually created and returned by :meth:`Scheduler.every`
method, which also defines its `interval`.
"""
def __init__(self, interval, scheduler=None):
    self.interval = interval  # pause interval * unit between runs
    self.latest = None  # upper limit to the interval
    self.job_func = None  # the job job_func to run
    self.unit = None  # time units, e.g. 'minutes', 'hours', ...
    self.at_time = None  # optional time at which this job runs
    self.last_run = None  # datetime of the last run
    self.next_run = None  # datetime of the next run
    self.period = None  # timedelta between runs, only valid for
    self.start_day = None  # Specific day of the week to start on
    self.tags = set()  # unique set of tags for the job
    self.scheduler = scheduler  # scheduler to register with

def __lt__(self, other):
    """
    PeriodicJobs are sortable based on the scheduled time they
    run next.
    """
```

```
    return self.next_run < other.next_run


def __str__(self):
    return (
        "Job(interval={}, "
        "unit={}, "
        "do={}, "
        "args={}, "
        "kwargs={})"
    ).format(self.interval,
             self.unit,
             self.job_func.__name__,
             self.job_func.args,
             self.job_func.keywords)


def __repr__(self):
    def format_time(t):
        return t.strftime('%Y-%m-%d %H:%M:%S') if t else '[never]'


    def is_repr(j):
        return not isinstance(j, Job)


    timestats = '(last run: %s, next run: %s)' % (
                format_time(self.last_run), format_time(self.next_run))


    if hasattr(self.job_func, '__name__'):
```

```
    job_func_name = self.job_func.__name__
else:
    job_func_name = repr(self.job_func)
args = [repr(x) if is_repr(x) else str(x) for x in self.job_func.args]
kwargs = ['%s=%s' % (k, repr(v))
          for k, v in self.job_func.keywords.items()]
call_repr = job_func_name + '(' + ', '.join(args + kwargs) + ')'

if self.at_time is not None:
    return 'Every %s %s at %s do %s %s' % (
            self.interval,
            self.unit[:-1] if self.interval == 1 else self.unit,
            self.at_time, call_repr, timestats)
else:
    fmt = (
        'Every %(interval)s ' +
        ('to %(latest)s ' if self.latest is not None else '') +
        '%(unit)s do %(call_repr)s %(timestats)s'
    )

    return fmt % dict(
        interval=self.interval,
        latest=self.latest,
        unit=(self.unit[:-1] if self.interval == 1 else self.unit),
        call_repr=call_repr,
        timestats=timestats
```

```
        )


@property
def second(self):
    if self.interval != 1:
        raise IntervalError('Use seconds instead of second')
    return self.seconds


@property
def seconds(self):
    self.unit = 'seconds'
    return self


@property
def minute(self):
    if self.interval != 1:
        raise IntervalError('Use minutes instead of minute')
    return self.minutes


@property
def minutes(self):
    self.unit = 'minutes'
    return self


@property
def hour(self):
```

```
    if self.interval != 1:
        raise IntervalError('Use hours instead of hour')
    return self.hours


@property
def hours(self):
    self.unit = 'hours'
    return self


@property
def day(self):
    if self.interval != 1:
        raise IntervalError('Use days instead of day')
    return self.days


@property
def days(self):
    self.unit = 'days'
    return self


@property
def week(self):
    if self.interval != 1:
        raise IntervalError('Use weeks instead of week')
    return self.weeks
```

```
@property
def weeks(self):
    self.unit = 'weeks'
    return self


@property
def monday(self):
    if self.interval != 1:
        raise IntervalError('Use mondays instead of monday')
    self.start_day = 'monday'
    return self.weeks


@property
def tuesday(self):
    if self.interval != 1:
        raise IntervalError('Use tuesdays instead of tuesday')
    self.start_day = 'tuesday'
    return self.weeks


@property
def wednesday(self):
    if self.interval != 1:
        raise IntervalError('Use wednesdays instead of wednesday')
    self.start_day = 'wednesday'
    return self.weeks
```

```
@property
def thursday(self):
    if self.interval != 1:
        raise IntervalError('Use thursdays instead of thursday')
    self.start_day = 'thursday'
    return self.weeks

@property
def friday(self):
    if self.interval != 1:
        raise IntervalError('Use fridays instead of friday')
    self.start_day = 'friday'
    return self.weeks

@property
def saturday(self):
    if self.interval != 1:
        raise IntervalError('Use saturdays instead of saturday')
    self.start_day = 'saturday'
    return self.weeks

@property
def sunday(self):
    if self.interval != 1:
        raise IntervalError('Use sundays instead of sunday')
    self.start_day = 'sunday'
```

```
    return self.weeks


def tag(self, *tags):
    """
    Tags the job with one or more unique indentifiers.

    Tags must be hashable. Duplicate tags are discarded.

    :param tags: A unique list of ``Hashable`` tags.
    :return: The invoked job instance
    """
    if not all(isinstance(tag, Hashable) for tag in tags):
        raise TypeError('Tags must be hashable')
    self.tags.update(tags)
    return self


def at(self, time_str):
    """
    Specify a particular time that the job should be run at.

    :param time_str: A string in one of the following formats: `HH:MM:SS`,
        `HH:MM`,`:MM`, `:SS`. The format must make sense given how often
        the job is repeating; for example, a job that repeats every minute
        should not be given a string in the form `HH:MM:SS`. The difference
        between `:MM` and `:SS` is inferred from the selected time-unit
        (e.g. `every().hour.at(':30')` vs. `every().minute.at(':30')`).
```

```
:return: The invoked job instance
"""
if (self.unit not in ('days', 'hours', 'minutes')
        and not self.start_day):
    raise ScheduleValueError('Invalid unit')
if not isinstance(time_str, str):
    raise TypeError('at() should be passed a string')
if self.unit == 'days' or self.start_day:
    if not re.match(r'^([0-2]\d:)?[0-5]\d:[0-5]\d$', time_str):
        raise ScheduleValueError('Invalid time format')
if self.unit == 'hours':
    if not re.match(r'^([0-5]\d)?:[0-5]\d$', time_str):
        raise ScheduleValueError(('Invalid time format for'
                                  ' an hourly job'))
if self.unit == 'minutes':
    if not re.match(r'^:[0-5]\d$', time_str):
        raise ScheduleValueError(('Invalid time format for'
                                  ' a minutely job'))
time_values = time_str.split(':')
if len(time_values) == 3:
    hour, minute, second = time_values
elif len(time_values) == 2 and self.unit == 'minutes':
    hour = 0
    minute = 0
    _, second = time_values
else:
```

```
        hour, minute = time_values
        second = 0
    if self.unit == 'days' or self.start_day:
        hour = int(hour)
        if not (0 <= hour <= 23):
            raise ScheduleValueError('Invalid number of hours')
    elif self.unit == 'hours':
        hour = 0
    elif self.unit == 'minutes':
        hour = 0
        minute = 0
    minute = int(minute)
    second = int(second)
    self.at_time = datetime.time(hour, minute, second)
    return self


def to(self, latest):
    """
    Schedule the job to run at an irregular (randomized) interval.

    The job's interval will randomly vary from the value given
    to  `every` to `latest`. The range defined is inclusive on
    both ends. For example, `every(A).to(B).seconds` executes
    the job function every N seconds such that A <= N <= B.

    :param latest: Maximum interval between randomized job runs
```

```
    :return: The invoked job instance
    """
    self.latest = latest
    return self


def do(self, job_func, *args, **kwargs):
    """
    Specifies the job_func that should be called every time the
    job runs.

    Any additional arguments are passed on to job_func when
    the job runs.

    :param job_func: The function to be scheduled
    :return: The invoked job instance
    """
    self.job_func = functools.partial(job_func, *args, **kwargs)
    try:
        functools.update_wrapper(self.job_func, job_func)
    except AttributeError:
        # job_funcs already wrapped by functools.partial won't have
        # __name__, __module__ or __doc__ and the update_wrapper()
        # call will fail.
        pass
    self._schedule_next_run()
    self.scheduler.jobs.append(self)
```

```
    return self


@property
def should_run(self):
    """
    :return: ``True`` if the job should be run now.
    """
    return datetime.datetime.now() >= self.next_run


def run(self):
    """
    Run the job and immediately reschedule it.

    :return: The return value returned by the `job_func`
    """
    logger.info('Running job %s', self)
    ret = self.job_func()
    self.last_run = datetime.datetime.now()
    self._schedule_next_run()
    return ret


def _schedule_next_run(self):
    """
    Compute the instant when this job should run next.
    """
    if self.unit not in ('seconds', 'minutes', 'hours', 'days', 'weeks'):
```

```
    raise ScheduleValueError('Invalid unit')


if self.latest is not None:
    if not (self.latest >= self.interval):
        raise ScheduleError('`latest` is greater than `interval`')
    interval = random.randint(self.interval, self.latest)
else:
    interval = self.interval


self.period = datetime.timedelta(**{self.unit: interval})
self.next_run = datetime.datetime.now() + self.period
if self.start_day is not None:
    if self.unit != 'weeks':
        raise ScheduleValueError('`unit` should be 'weeks'')
    weekdays = (
        'monday',
        'tuesday',
        'wednesday',
        'thursday',
        'friday',
        'saturday',
        'sunday'
    )
    if self.start_day not in weekdays:
        raise ScheduleValueError('Invalid start day')
    weekday = weekdays.index(self.start_day)
```

```
    days_ahead = weekday - self.next_run.weekday()
    if days_ahead <= 0:  # Target day already happened this week
        days_ahead += 7
    self.next_run += datetime.timedelta(days_ahead) - self.period
if self.at_time is not None:
    if (self.unit not in ('days', 'hours', 'minutes')
            and self.start_day is None):
        raise ScheduleValueError(('Invalid unit without'
                                  ' specifying start day'))
    kwargs = {
        'second': self.at_time.second,
        'microsecond': 0
    }
    if self.unit == 'days' or self.start_day is not None:
        kwargs['hour'] = self.at_time.hour
    if self.unit in ['days', 'hours'] or self.start_day is not None:
        kwargs['minute'] = self.at_time.minute
    self.next_run = self.next_run.replace(**kwargs)
    # If we are running for the first time, make sure we run
    # at the specified time *today* (or *this hour*) as well
    if not self.last_run:
        now = datetime.datetime.now()
        if (self.unit == 'days' and self.at_time > now.time() and
                self.interval == 1):
            self.next_run = self.next_run - datetime.timedelta(days=1)
        elif self.unit == 'hours' \
```

```
                    and self.at_time.minute > now.minute \
                    or (self.at_time.minute == now.minute
                        and self.at_time.second > now.second):
                self.next_run = self.next_run - datetime.timedelta(hours=1)
            elif self.unit == 'minutes' \
                    and self.at_time.second > now.second:
                self.next_run = self.next_run - \
                                datetime.timedelta(minutes=1)
        if self.start_day is not None and self.at_time is not None:
            # Let's see if we will still make that time we specified today
            if (self.next_run - datetime.datetime.now()).days >= 7:
                self.next_run -= self.period


# The following methods are shortcuts for not having to
# create a Scheduler instance:

#: Default :class:`Scheduler <Scheduler>` object
default_scheduler = Scheduler()

#: Default :class:`Jobs <Job>` list
jobs = default_scheduler.jobs  # todo: should this be a copy, e.g. jobs()?


def every(interval=1):
    """Calls :meth:`every <Scheduler.every>` on the
```

```
    :data:`default scheduler instance <default_scheduler>`.
    """
    return default_scheduler.every(interval)


def run_pending():
    """Calls :meth:`run_pending <Scheduler.run_pending>` on the
    :data:`default scheduler instance <default_scheduler>`.
    """
    default_scheduler.run_pending()


def run_all(delay_seconds=0):
    """Calls :meth:`run_all <Scheduler.run_all>` on the
    :data:`default scheduler instance <default_scheduler>`.
    """
    default_scheduler.run_all(delay_seconds=delay_seconds)


def clear(tag=None):
    """Calls :meth:`clear <Scheduler.clear>` on the
    :data:`default scheduler instance <default_scheduler>`.
    """
    default_scheduler.clear(tag)
```

```
def cancel_job(job):
    """Calls :meth:`cancel_job <Scheduler.cancel_job>` on the
    :data:`default scheduler instance <default_scheduler>`.
    """
    default_scheduler.cancel_job(job)


def next_run():
    """Calls :meth:`next_run <Scheduler.next_run>` on the
    :data:`default scheduler instance <default_scheduler>`.
    """
    return default_scheduler.next_run


def idle_seconds():
    """Calls :meth:`idle_seconds <Scheduler.idle_seconds>` on the
    :data:`default scheduler instance <default_scheduler>`.
    """
    return default_scheduler.idle_seconds
```