Code Review are a great tool and you should do it often. [Best Practices for Code Review](https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/) gives a great summary including benifits and top 10 tips. Checklist are great tool, [NASA’s Space Shuttle Operational Flight Rules](https://www.jsc.nasa.gov/news/columbia/fr_generic.pdf) is a great example. Current checklist is heavily inspired from [this Checklist](https://www.liberty.edu/media/1414/%5B6401%5Dcode_review_checklist.pdf). Python specific elements are added to it which are inspired from [Pycon 2016 Talk](https://www.youtube.com/watch?v=D_6ybDcU5gc). Go ahead and use this checklist for:

* Guide while starting new project
* Refactoring an existing project
* Rewriting a legacy project

**Structure**

* Is the code PEP-8 compliant?
* Are there any uncalled or unneeded procedures or unreachable code?
* Can a code be replaced by standard library calls?
* Are there any blocks of codes that be made into single function?
* Are there any Magic Numbers or Constants?
* Does the code have any logical orgnaization or its all flat?
* Are there any complex modules that can be broken down into smaller chunks?

**Documentation**

* Is there any documentation?
* Does the code explain what main modeules and entities are?
* Does the code have any comments?
* Are comments in these code meaningful?
* Do we have key classes/functions documented using right? {Read more on [PEP257](https://www.python.org/dev/peps/pep-0257/) for more details}
* Do we have a README.md?
* Do we have a WORKLOG.md?
* Do we have a TESTING.md?
* Are steps for setting up the project documented in README.md?
* Do we have a documentation of Key Models in README.md?
* Do we have documentation on key management commands?
* Do we have documentation of Key Roles in TESTING.md?
* Do we have a [Permission Matrix](https://support.procore.com/references/user-permissions-matrix-web)?
* Do we have a list of Critical Workflows for application in TESTING.md?
* Do we have list of Browsers with specific versions mentioned that we will be testing in TESTING.md?
* Do we have list of Mobile devices on which we will be doing testing in TESTING.md?

**Variables**

* Does the code have abstract variables names as i,j and k?
* Does the code have clear and explicit names?
* Are there any redundant or unused variables?

**Arithmetic Operations**

* Does the code have any blocks that are heavy in computations?
* Does code avoid floating point computations for comparision?
* Are computation calculations taken care of divide by zero errors?
* Are there unit-tests for computation logic?

**Loops, Branches, Function and Classes**

* Are all loops and branches properly nested?
* Are most common cases in IF-ELSEIF block tested first?
* Are all cases covered in IF-ELSEIF-ELSE blocks?
* Are loops terminating on valid conditions?
* Can any statement that is within the loop placed outside the loop?
* Can blocks of IF-ELSE statement be named into function?
* Can blocks of functions be named into a class?
* Do we have a Class with just one single public function?
* Can blocks of function be moved into a module/external file?
* Does the Function/Class follow [Single Responsibility Principle](https://en.wikipedia.org/wiki/Single_responsibility_principle)?

**Defensive programming**

* Do we have any blocks of code that should have TRY-EXCEPT blocks but don’t have them ?
* Are there any errors that are silent?
* Do we have test case or unit tests written?

**Django Specific Checks**

* Do we have segregated settings {local.py, dev.py, prodction.py and base.py}?
* Do we have updated requirements.txt?
* Do we use PostgresSQL as default DB?
* Do we have one-off scripts that can be extracted into a Django Management Command?
* Do we have commented urls in urls.py?
* Are we using the right patterns for URLs? {[List of Useful URL Patterns](https://simpleisbetterthancomplex.com/references/2016/10/10/url-patterns.html) is a great reference}
* Are we importing specific Classes/Functions from modules from accounts\_manager.models import AccountsManager as opposed to from accounts\_manager.models import \*?
* Do we have singular application and model names? {E.g. accounts\_manager as opposed to accountsmanager}
* Are we storing contents of email outside code? {E.g. store contents of emails in templates/emails/}. Use [render-string](https://docs.djangoproject.com/en/1.10/topics/templates/#django.template.loader.render_to_string) for substitions
* Do we have Skinny Views and Fat Models? {[Reference Blogpost](https://hackerfall.com/story/fat-models--a-django-code-organization-strategy) explains with detail}
* Are we using standard packages like [django-haystack](http://haystacksearch.org/) as opposed to writing custom low level queries to search indexer?
* Do we have following integrations from day one?
  + [Sentry](https://sentry.io/welcome/) for error tracking
  + [HotJar](https://www.hotjar.com/) for ux testing
  + [Google Analytics](https://analytics.google.com/) for analytics
* Are we profiling code using [django-debug-toolbar](http://django-debug-toolbar.readthedocs.io/en/stable/), do know how many queries are getting fired on most pages?
* Are we using redis for session backend? {[Reference Blogpost](http://michal.karzynski.pl/blog/2013/07/14/using-redis-as-django-session-store-and-cache-backend/)}
* Do we have simple template tags? {Ideally no DB calls in template tags}
* Are we minifing all our static assets using [django-compressor](https://django-compressor.readthedocs.io/en/latest/) ?
* Do we have our static on CDN?
* Do we have deployment scripted using [Ansible](https://www.ansible.com/) ?