# PYTHON BEST PRACTICES - TIPS AND TRICKS...

# Full Stack Ascent

* Sorry no memes because I decided to put the whole lesson in one python file for some reason – but here is an ascii art one.
* Fluid and wide ranging – it’s not about building out a particular python skill but I thought it would be good to share some good coding practices that will get your code looking legit and sexy.
* These concepts and pointers should help you build good looking python code in whatever context you’re writing it – and some of this stuff should carry over to other languages as well.
* Thank you to those who sent me some of their code. This was really helpful to have an understanding of areas of improvement.
* The best way to get better is to write code and have someone look at it and tell you how to improve it tbh – also you can ask copilot – but ask it to explain it to you if you do so you get better.
* How people get better is by proper code reviews.
* So I’m hoping that lots of us are at a stage where we can get code that works and that’s 90% of it. So well done if you can do that.
* In my mind the next step in improving code is taking that code that works and making sure it is written with what we’re calling here Python Best Practices.
* What does this mean and why do we want to do it?
  + Writing with performance in mind.
    - There are a thousand ways to solve any one problem in code.
    - Challenge 1 is usually finding just one of these ways.
    - And Challenge 2 is working out (one of) the most performant way of writing it.
  + Security
  + Being nice to your future self.
  + Being nice to someone else.
    - Extending existing functionality
    - Refactoring existing functionality

# - PEP 8

* We won’t dwell on it because it’s quite boring but I thought I should mention PEP 8 - Python Enhancement Proposal
  + Comes from the Python Community
  + If interested in specifics have a read through
  + Guidelines more than Hard rules
  + **Naming Conventions**: Specifies how to name variables, functions, and classes for consistency.
  + **Spacing Around Operators**: Describes how to format spaces around operators (e.g., a + b).
  + **Imports**: Guidelines on how to structure import statements like the order.
  + **Blank Lines**: Recommends how many blank lines to use for readability between functions, classes, and top-level code.
  + **Line Length**: Suggests limiting lines to 79 characters for readability ( this is contentious 120 is usually fine)
  + **Linters and Formatters**: Tools to automatically check and format your code to comply with PEP 8 – e.g. Black, Flake8, Ruff, isort.

# - Readability Tips

* Speaking of Making Readable Python so you and others can understand what the hell is going on in your code. Let’s talk through a few tips!

# - Descriptive Naming

* + Clear descriptive names for variables, functions and classes
  + Calculate\_area is better than calc\_a

# - Comments and Docstrings

Use comments to explain really complex sections of your code.

Comments should be used sparingly and only when necessary. But don’t overdo it - your code should be self-explanatory.

If you find yourself writing loads of comments see this as a warning sign that maybe your code isn’t readable.

For functions and classes, write docstrings to describe what they do, their parameters, and return values.

# - Constants over Magic Numbers and Strings

* You shouldn’t directly use numbers and strings you should use constants – in Python they are variables with capital letters

# - Avoid Deep Nesting

There are lots of ways to refactor this code to make it more readable.

We will cover some of these in the next sections.

So when we get to the bits on modularity and then on control flow remember how ugly this is.

# - Functions and Modularity

# - Single Responsibility Principle

# - DRY (Don’t Repeat Yourself)

# - Modules to break down large files

# - Cool Stuff

# - List Comprehensions

# - Enumerate

# - Zip

# - Generators / Lazy Iteration

# - Control Flow

# - Ternaary Operators

# - == vs is

# - Type Hinting

# - Unpacking