**100 days of code(Python and ML)**

100 Days of Code is a license to experiment as a developer

**Aim – get hands on practice and get Tensorflow certification**

**Rules -**

* code 1 hour every day
* Doesn't include tutorials and concept learning
* No copy-pasting of code. All hand-typed
* Post daily **goal and progress** on twitter/Discord server
* Keep a journal - Github
* If something ultra urgent comes up- you can miss a single day/week in a series. Miss more than that - add that to beyond 100 days.
* Separate 1-2 hours for concepts building or understanding basics. So assume **3 hours per day** minimum requirement.

**Tools to use -**

Tensorflow, Keras, Google Colab, Pycharm, Python

**Topics to learn**

Python and Tensorflow

ML model

DL models(vision and language)

**Resources to study**

Tensorflow specialization - Coursera

**Daily Plan-**

**Python basics**

1. Pytest code
2. Error handling
3. ~~Slicing text and strings - Re(Regex) in python~~
4. ~~Web images scrapping framework~~
5. ~~Web text scrapping framework~~
6. ~~Github and git~~
7. Pycharm basics – creating virtual environment
8. Pycharm advanced – using tensorboard and Tensorflow extensions

**Tensorflow**

1. Tensorflow and its components
2. Tensorboard course asssignments – 1
3. Tensorboard course asssignments - 2
4. Weights and biases
5. Deploying code from Jupyter notebook
6. Visual coding in Deep learning
7. Feature engineering pipeline for Images
8. Feature engineering pipeline for text
9. Tensorboard intergration
10. Tensorboard/graph based – detecting overfitting and underfitting

**Building up on CNN and Deep learning skiils**

1. CNNs code in tensorflow
2. K-cross validation
3. Looking into filters of CNN
4. GAN

**Machine learning in tensorflow**

1. Xgboost
2. Naive bias classifier
3. Support vector machine(SVM)

**Time series prediction**

1. Support vector machine(SVM)
2. Time series models
3. Sequence models
4. Prediction models
5. RNNs and CNNs for time series, sequence and forecasting models
6. Forecasting.
7. Identify and compensate for sequence bias.

**Natual language processing**

1. RNNS, LSTMs, for text
2. Binary categorization for text
3. Multi-class categorization in text
4. NLP introduction course asssignments– 1
5. NLP introduction course asssignments- 2
6. GRUs for text
7. CNNs in models that work with text.
8. Text preprocessing
9. Chat bot in tensorflow
10. Twitter bot – basic - 1
11. Twitter bot – good - 2
12. Amazon review analysis

**CV DL advanced**

1. Segmentation unet from sratch
2. Image based emotional analysis

**Kaggle competitions**

1. Mimic best Pneumonia detection solution
2. Mimic best Opacity detection solution

**NLP projects**

1. Text based sentiment detection
2. Using transfer learning in text
3. Chunking and chinking in NLP

**NLP advanced projects**

1. BERT, GPT-2 , GPT-3
2. Autocompelete sentence - 1
3. Autocompelete sentence -2
4. Autocompelete sentence – 3
5. Pre-written scipts for NLP - 1
6. Pre-written scipts for NLP - 2
7. Pre-written scipts for CNNs
8. Pre-written scipts for Time series

What these projects look like will vary from developer to developer, but a few key points to remember:

* Implementing more technologies and frameworks will generally be more complex and challenging.
* Unfamiliar technologies often have a hidden cost that you should account for when creating a timeline. Do not underestimate the likelihood of unforeseen issues and complexity.
* More popular frameworks and tools will have better documentation and more example projects for you to explore. Tapping into a strong developer community will save you time and energy.

Three additional tools that make your 100 Days of Code even better:

* Journal: Pick a medium to capture your thoughts during the 100 Days of Code challenge. You can create a blog, repository, or document to keep track of your progress. Kallaway, the creator of the 100 Days of Code challenge, created a journal template [100-days-of-code](https://github.com/kallaway/100-days-of-code). Software also created a clean and easy-to-use [template](https://github.com/geoffstevens8/100-days-of-code) based on some of the best practices in this guide that any developer can quickly deploy.
* Timer or tracker: Find a tool that can track how much you code each day so that you can ensure you code for at least an hour per day. You can use a simple stopwatch or you can opt for a more robust tracker that keeps a historical log of your coding time. see your progress over time.
* Project manager: You will need to juggle a lot of tasks, so find a tool that can help you regain context quickly and make the most of your hour each day.