In this doc some MUST HAVES for successful passing of labs will be provided.

**0.**  **Install ANY Linux Distribution Kit (Ubuntu, Arch Linux, Debian) or MacOS either on VM or as a second system. Do all labs on this OS.**

1. **Version Control System  
   a)** You are (if you still don’t have) to create an account on GitHub.  
   **b)** On this account you are to create a repo called, say, “Python labs” or smth of the kind.  
   **c)** For each lab you are to create a separate directrory named “Lab1”, “Lab2”, etc.  
   **d)** Each step of developing your lab must be commited and pushed to the repo. Labs done with one/two/three commits won’t be passed. Name your commits correctly, name must illustrate what is done in the commit but please be laconic.  
   Ex.: “added tests for my\_module.py” - **good**/ “in this commit i added some code to test smth” - **bad.  
   e)** Each lab must be developed in a separate branch named after the lab. Before the attempt of passing the lab PR to main branch must be created.  
   **f)** In the repo I shouldn't see anything like .pycache or .idea files, use .gitignore to check it. “venv” directory shouldn’t be there either. Instead of it there must be a “requirements.txt” file with all dependencies you need for your code.
2. **Project structure  
   a)** For each lab you should have a separate directory named “Lab1”, “Lab2”, etc. and for each lab you should create its own “venv”. Exception is for connected labs (one project divided into several labs). Labs without “venv” (virtual environment) are not allowed.  
   **b)** Any libraries that do the task of the lab instead of you **are BANNED**!   
   **c)** For code writing you may use any editors or IDE you want (VSCode, PyCharm, Notepad++, Atom, Vim, etc.).  
   **d)** Your code can’t be in one file: you are to separate it into modules/files where each file has its own logic.  
   Ex.: main.py - file from which the program starts, input and output happen, other modules are called;  
   utilities.py/helpers.py - file where all small helpers and utilities are implemented;  
   constants.py - file where constants are kept;  
   distributions.py - file where declarations of some distributions classes are kept (this is a specific example if you have a task to implement distributions methods on your own) .  
   **The code either in one file or not logically separated is BANNED!**
3. **Code rules  
   a)** The code must be clean: try to find the best algorithm and the most pythonic way to create it. It will be rather difficult first but if you try you will see how beautiful Python can be.  
   **b)** Any logical block(if-else statement, for/while statement, return statement) inside the code must be separated by blank lines just to be more readable.  
   **c)** from … import \* **is BANNED!** Please import only needed methods from other modules! Please show the full way(in terms of the project) to the module from which you import   
   **d)** The examples of rules of variables, methods, functions and classes naming, logical blocks and statements separating are provided in the screenshot below  
     
   Please try to avoid “magic numbers” - don’t hardcode pure strings of numbers, **put them into constants of variables instead!**
4. **How to Pass the Lab  
   0) THE CODE IS YOURS!!!!  
   a)** Code check - your code works both on correct and incorrect values!  
   **b)** Rules check - all the rules above (1-3) are done up to the lab!  
   **c)** Understanding check - you understand everything inside your code!  
   **d)** Theory check - you know the theory about the lab!

**e)** CONGRATULATIONS! The lab is over! Now you can start the next one!