Introduction to Competitive Programming using Python



*\* In this new series of periodical publication, SnB Political and Economic Research Institute is introducing step-by-step educational contents on data analysis and machine-learning using Python interpreter.*

Data Analysis and Machine-Learning using Python

Chapter 1: Introduction and Environment Setting for Competitive Programming

To our readers and subscribers of SnB PERI who are interested in econometrics, financial engineering, statistics, or various other research branches using data analysis, I strongly recommend to learn coding (Python and R in particular) as programming allows for a considerable extension in terms of ‘Types’ and ‘Volumes’ of data you can process with.

Let’s say, for instance, you want to develop an automatized model for stock or crypto-currency investment (i.e., algorithm design and automation). Or, you might want to conduct a research on predicting future inflation rates based on survey data (i.e., text analysis or sentiment analysis). You may also want to create and process your “own” dataset in case of facing difficulties in finding appropriate data (crawling and data-mining). Last but not least, you may want to improve the statistical accuracy of your model via artificial intelligence (logistic regression and machine-learning).

The common prerequisite for conducting all these seemingly-complex tasks is to know “how to code.” Of course, some of the topics outlined earlier requires mathematical and statistical understandings (which will also be covered in later chapters of this series). But there is no doubt that coding skills are the very common molecules that become the basis of all these exciting spheres, which is why it cannot be underscored enough that coding skills are the most important and basic element for exploring these fields.

With all that being said, I would like to start by outlining the distinct features of this series and just one simple advice for coding.

***1. Learn with your hands (not just with your head)***

The most common mistake that beginners to programming make is trying to process the whole algorithm logics and procedures in their head.

The peculiar feature of programming is that “it always makes sense” and sufficiently comprehensible, theoretically. When you hear an explanation or look at the completed codes, it might feel like you understood it all and can replicate later on.

However, the real hard part about coding is usually the simplest things: indentations, syntax, types, etc.

The only way to get better at coding is by typing with hands and frequently seeing the output line by line, and to learn from repeating errors. Someone might make less mistakes on indentations, but more on syntax. Others might be exactly the opposite. You may never know before you “actually type with your hands” and see the results frequently.

***2. This first chapter is going to be strictly focused on actual algorithm cases and practices***

In connection with the previous line of advice, this chapter is going to be strictly based on actual cases, problem-solving, and simple algorithm-making.

As emphasized earlier, less on theory and more on actual typing; the chapter is going to assume that the readers have the basic understanding regarding the basic grammars used in python and REPL (Read, Evaluation, Print, Loop) mechanism of Python interpreter. But don’t worry. The problems and cases are going to be presented step-by-step based on difficulties, and the answers are going to be included together.

In case you might need to review the grammars, I recommend referencing the official Python website manual:

<https://docs.python.org/3/tutorial/index.html>

You can also convert the page into your mother tongue at the left-top side of the webpage.

***3. Competitive Programming***

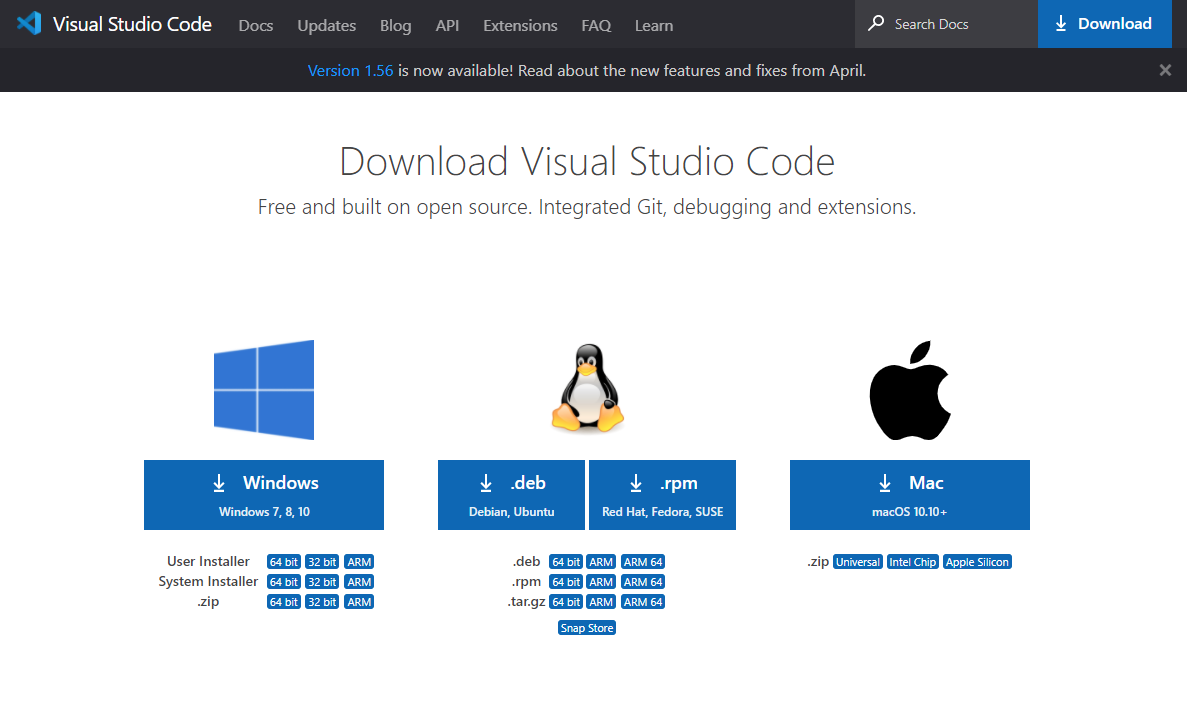
With the previous two points being said, Thus, purpose of the first chapter is to expose the readers to algorithm-making and problem-solving environment.

Basically, the problem definition and input value are going to be presented with the target output. Your goal is to practice by creating an algorithm code that generates the output value. Again, do not worry too much. As previously said, the problems and cases are going to be presented step-by-step based on difficulty levels, and the sample answers are going to be included together. But don’t forget, type the actual codes with your hands even after you see the answer, and check the output values in your terminal.

Let us begin by setting the environment for competitive programming. For practice, I recommend using Visual Studio Code as your Python interpreter.

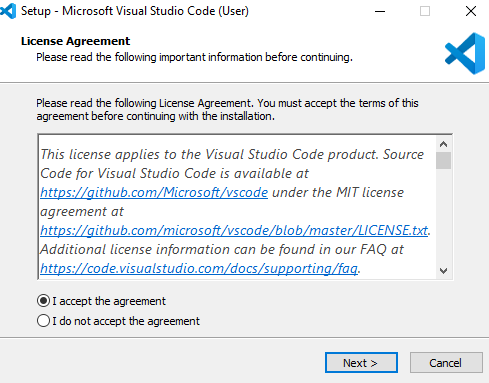
***1) Download Visual Studio Code and Complete Installation***

Download URL: <https://code.visualstudio.com/download>



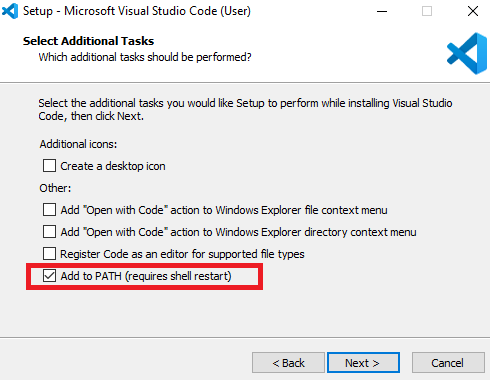
Download and launch the exe file.

Accept the agreement, and click on “Next”

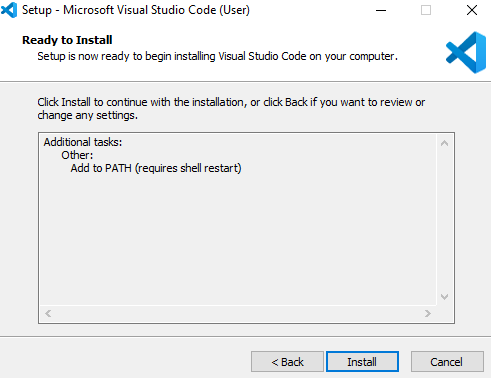


Do **not** forget to check ***“Add to Path”*** icon.

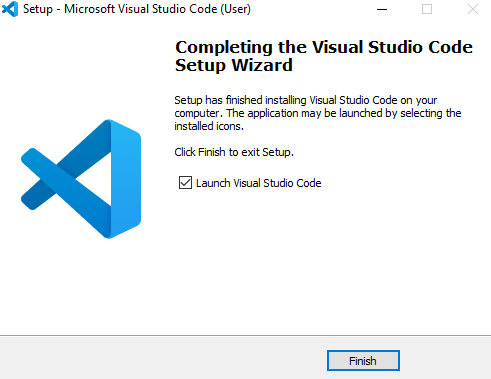
Click on “Next” to proceed.



Click on ***“Install”***.



Finish Setup and launch Visual Studio Code.

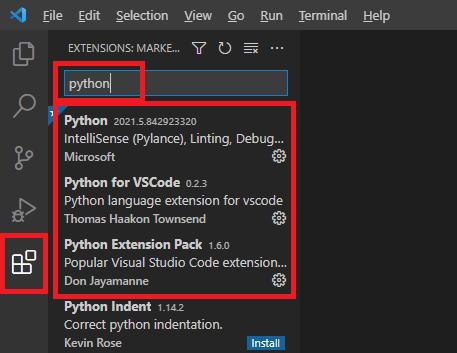


***2) Extensions Setting***

After launching VSCode, Click ***“Extensions”*** at the left menu.

As for now, you will only need ***“Python”*** by Microsoft, ***“Python for VSCode”*** by Thomas Haakon Townsend, and ***“Python Extension Pack”*** by Don Jayamanne.

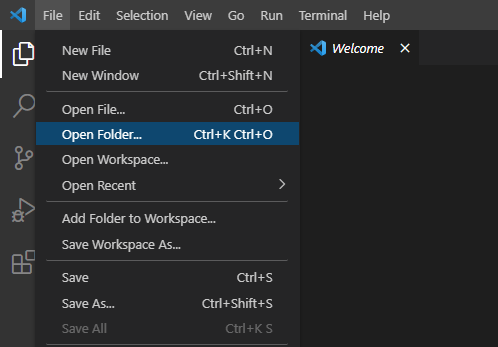
Proceed to Installation of the required extensions after typing the above three extensions.



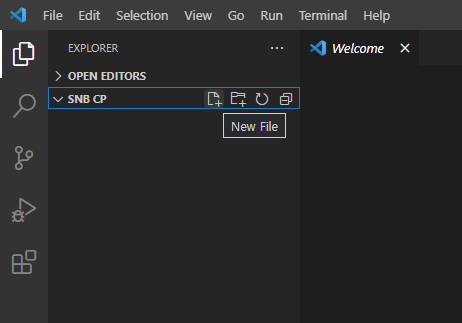
***3) VSCode Environment Settings for Competitive Programming***

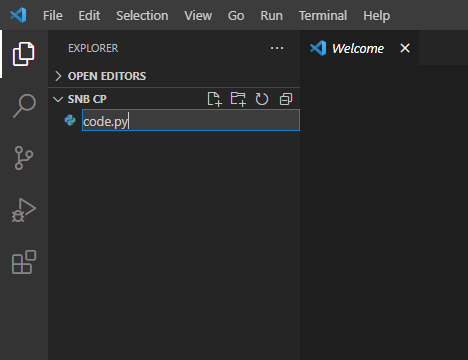
Create a new empty folder on desktop.

Click “File” -> “Open Folder” and select the created folder.

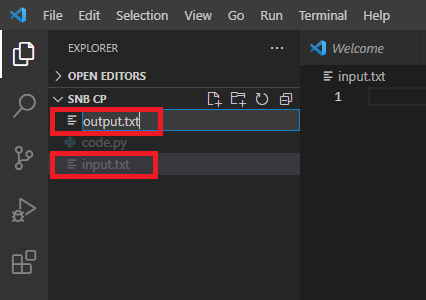


After selecting your folder, create a new Python file (\*.py) by clicking the icon.

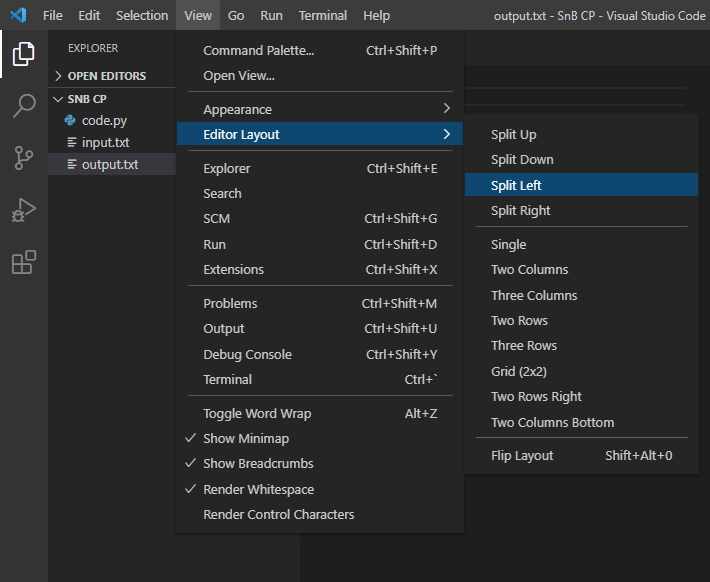




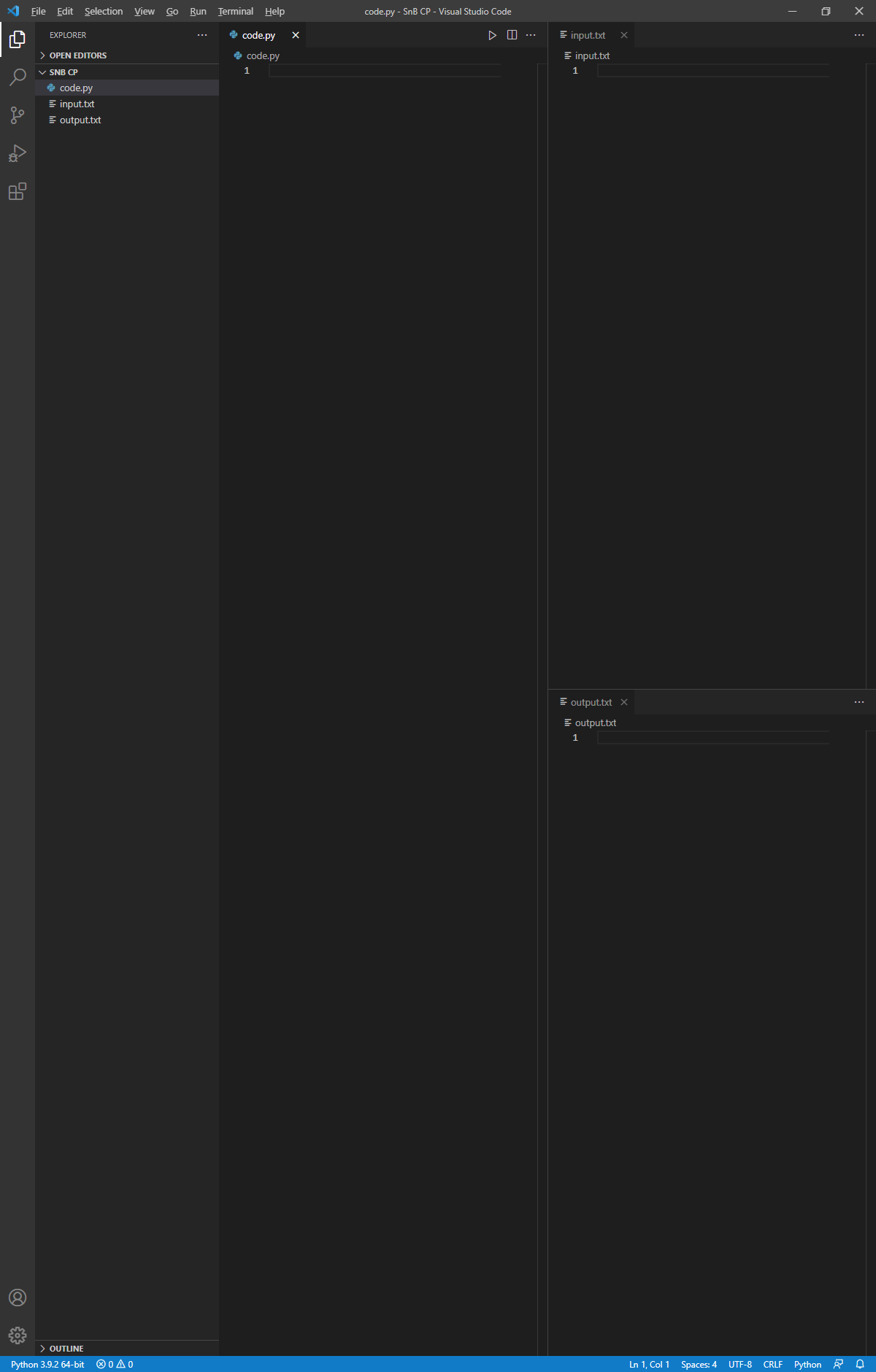
Now, create two new txt files (\*.txt) within your folder as your input and output displayers.



Now that you have three files ready, split the screens for convenient view ***[“View” -> Editor Layout -> Split (left, down)]***



The final editor layout would look like this.



Now to finalize the environment setting for competitive programming, you need to receive the input value from the input.txt file and display the executed output value on the output.txt file.

You can do this by importing sys and specifying stdin/stdout file (input.txt and output.txt, respectively) location and settings (“read” for input.txt, and “write” for output.txt).

Refer to the following sample codes and complete the environment setting.

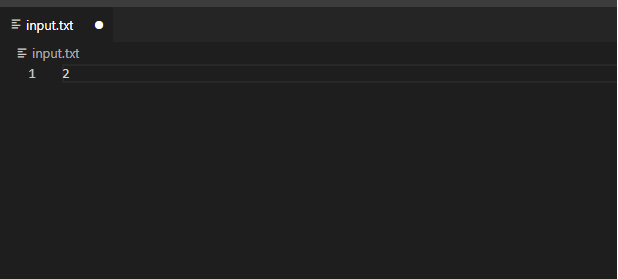
import sys

sys.stdout = open('C:\\Users\\Master\\Desktop\\SnB CP\\output.txt', 'w')

sys.stdin = open('C:\\Users\\Master\\Desktop\\SnB CP\\input.txt', 'r')

To check if all linkages are correctly completed, you can try by typing a simple code that processes the given input value, and see if the outputs are being displayed correctly on the output.txt file (Do NOT forget to save everything before you run the code).

For example, let’s type 2 as our input value.

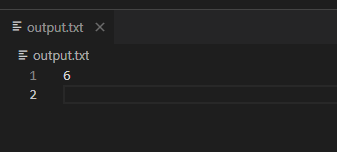


Save the input.txt file (ctrl + S)

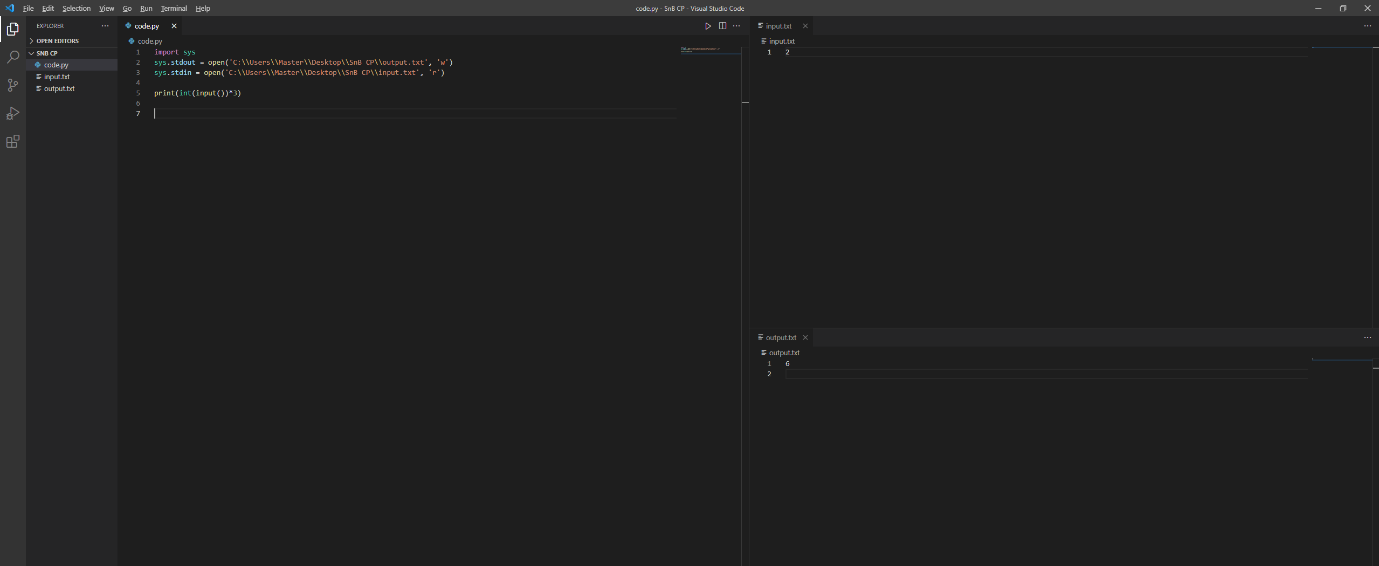
Now, let’s convert the input value into integer type, multiply it by 3, and print the output:

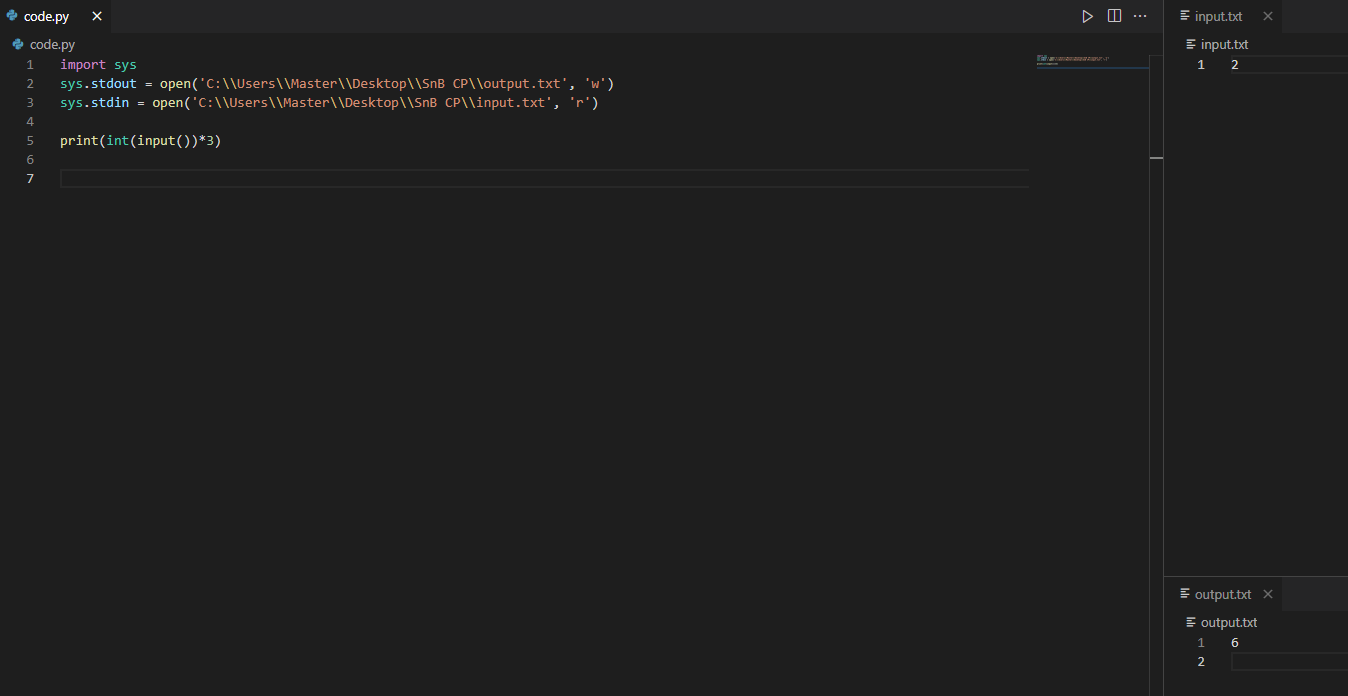
print(int(input())\*3)

If there’s no problem, the expected output value should be 6, displayed on the output.txt file, as follows.



The finalized view should look like this:





Upon the completion of these instructions, you are all ready and set to start competitive programming, i.e., problem-solving via coding and generating simple algorithms to draw desired output values.

Throughout the next sessions, sample problems are going to be presented step-by-step based on difficulty levels. Again, do not worry! Early-stage problems consist of simple formats which requires fundamental grammars. Plus, sample answers are going to be provided together with the problem.

Just one thing to remember. Code with your hands, not just with your head!