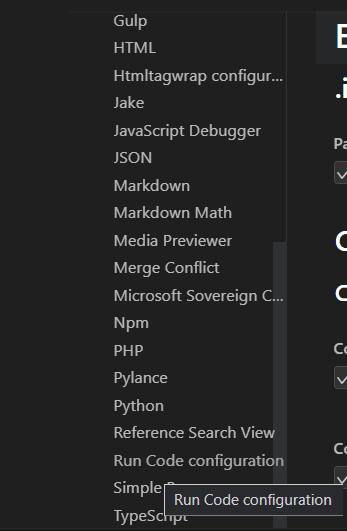
**DSA using Python + Leet code Exercises**

**1 - Environment Setup:**

* Install Python using Microsoft store.
* Install IDE like **Visual studio code.**
* I installed Python 3.12 from browser, Microsoft store, also from VS Extensions. I think from Microsoft store in more enough.
* For clean visible output – Install **Code Runner** from VS code extensions. Then
* Go to Settings

A screenshot of a computer

Description automatically generated

* Under **Extensions** – Choose **Run Code configuration**  
  
* Check this below box

A black background with white text

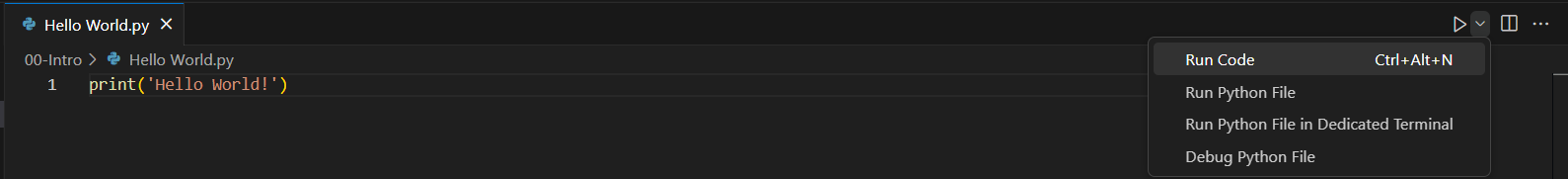
Description automatically generated

* Scroll down & Uncheck this below box

A black and white sign

Description automatically generated

* Now, Run the sample code using code runner or press Ctrl + Alt + N. which displays only output.



**2 – Big O (Important Topic for DSA)**

Please go through the supporting pdf

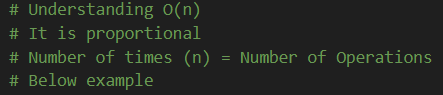


* Big O is way of comparing the efficiency of two codes mathematically. Generally, it can be measured using time complexity and space complexity of the code.
  + **Time Complexity –** It is the time takes to perform a piece of code or some set of tasks.
  + **Space Complexity –** It is the amount of memory taken to store a piece of code.
* We can mention these complexities by using the 3 Greek letters called
  + Omega – Ω - Best case scenario - also called Omega
  + Theta – Θ - Average case scenario - also called Theta
  + Omicron – O - Worst case scenario - **also called Big O(means worst case).**
* **Example – Consider a for loop of a list containing 7 numbers.**
  + Finding **1 be the best-case scenario**
  + Finding **7 be the worst-case scenario**

A screenshot of a phone

Description automatically generated

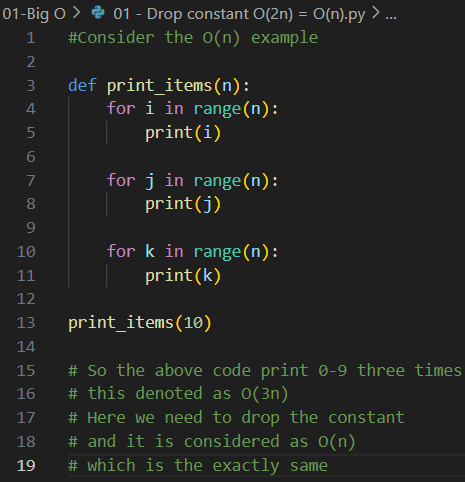
* **O(n)** 
  + It is more efficient or least efficient.
  + **A black screen with white text

    Description automatically generated **
  + The number of operations = number of times (n)
  + Ex: The function executes 10 times and n = 10.
  + In below diagram – **x axis = n; y axis = number of operations.**

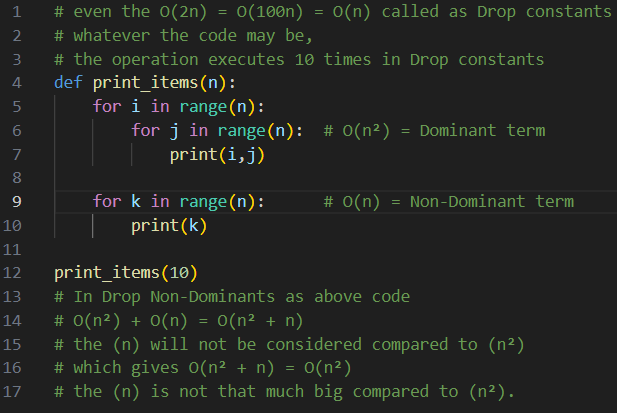
**A green line on a black background

Description automatically generated**

* + O(n) is proportional
* **Drop constant**
  + Consider two same loops inside the same function or even infinite loop inside the same function as show below
  + **Ex: O(n) = O(2n) = O(3n) = O(100n)**

****

* + **n = 10; operations = 10 (entire function consider as one operation)**
* **O(n²) or O (n \* n):**
  + Consider the function same as above but, this time, we prefer nested for loops which 10 \* 10 = 100 operations, but n = 10;
  + n = 10 < operations = 100. So the graph will be :
  + **A green line on a black background

    Description automatically generated**
  + Where **x-axis = n=10; y axis = operations = 100.**
* **Drop Non-Dominants**
  + Consider the O(n**²**) +O(n). **which is O (n² + n).**
  + **Please do the comment lines below**
  + ****
  + Therefore: **O (n² + n) = O(n²)** .
* **O (1):**
  + It is the most optical solution in Big O.
  + It performs only **1 operation** whatever the **n** may be.
  + **n=1 or 100 or more**; **operation = 1.**

A screenshot of a computer

Description automatically generated

* + In below graph**, x-axis: n = 10; y-axis: operations = 1**

**A graph on a black background

Description automatically generated**

* **O ( log n)**