



Sublime Text

tutorialspoint
SIMPLY EASY LEARNING

www.tutorialspoint.com



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

About the Tutorial

Sublime Text editor is a sophisticated text editor which is widely used among developers. It includes wide features such as Syntax Highlight, Auto Indentation, File Type Recognition, Sidebar, Macros, Plug-in and Packages that make it easy for working with code base.

This tutorial gives you a comprehensive coverage of concepts of Sublime Text and makes you comfortable to use it in your software development projects.

Audience

The target audience of this tutorial are developers of JavaScript and Python. Web developers who are looking for suitable Text editor like IDE will also benefit from this tutorial. After the completion of this tutorial, you will have an in-depth knowledge of Sublime Text editor.

Prerequisites

Before you proceed with this tutorial, we assume that you have a basic understanding of usage of various Text editors like Visual Studio code, PyCharm for Python and Integrated Development Environment (IDE) like NetBeans. You will understand this tutorial better if you have a basic knowledge of text editors.

The current version of Sublime Text editor is 3.0 and is compatible with various operating systems like Windows, Linux and MacOS.

Copyright & Disclaimer

@Copyright 2015 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

About the Tutorial.....	i
Audience	i
Prerequisites	i
Copyright & Disclaimer.....	i
Table of Contents.....	ii
1. SUBLIME TEXT – INTRODUCTION.....	1
Why Sublime Text?	1
2. SUBLIME TEXT – INSTALLATION.....	2
Installation on Windows	3
Installation on Linux.....	6
Installation on OSX.....	6
3. SUBLIME TEXT – DATA DIRECTORY	7
4. SUBLIME TEXT – CREATING FIRST DOCUMENT.....	10
5. SUBLIME TEXT – EDITING FIRST TEXT DOCUMENT	13
Simultaneous Write-Up	13
Finding Occurrences	14
Appending Comments in Line.....	15
6. SUBLIME TEXT – PATTERNS OF CODE EDITING	16
Splitting the Selection into Lines	16
Wrapping Complete Paragraph in HTML File	16
Finding all Occurrences	17
7. SUBLIME TEXT – SUBLIME LINTER	18

Modes of Sublime Linter	19
8. SUBLIME TEXT – SHORTCUTS	20
Opening the Command Palette	20
Toggle Side Bar.....	20
Display Scope in Status Bar	21
Python Console	21
New Window	22
9. SUBLIME TEXT – SNIPPETS.....	23
Understanding Snippets	23
Creating First Snippet	24
Package Control Snippets	26
10. SUBLIME – MACROS.....	27
Recording a Macro	27
Playing a Macro.....	28
Saving a Macro.....	28
11. SUBLIME TEXT – KEY BINDINGS	29
Defining Key Bindings.....	30
12. SUBLIME TEXT – COLUMN SELECTION.....	31
Moving the Pointer to the Beginning of Code.....	31
Extending a Column	32
Trimming a column	32
Split a Block of Text	32
13. SUBLIME TEXT – INDENTATION	33
Importance of Indentation	33
Options for Indentation in Sublime Text	33

Benefits of Indentation	35
14. SUBLIME TEXT – BASE SETTINGS	36
Categories of Settings.....	37
15. SUBLIME TEXT – THEME MANAGEMENT.....	39
Installing and Applying Themes.....	39
Color Schemes.....	41
16. SUBLIME TEXT – UNDERSTANDING VINTAGE MODE	43
Understanding vi editor	43
Enabling Vintage Mode	43
17. SUBLIME TEXT – VINTAGE COMMANDS	45
Vintageous Mode.....	45
18. SUBLIME TEXT – TESTING JAVASCRIPT	47
JsFormat.....	47
JSHint	48
JavaScriptNext	48
19. SUBLIME TEXT – TESTING PYTHON CODE.....	50
Installation of UnitTest.....	50
20. SUBLIME TEXT – SPELL CHECK	53
Dictionaries	53
Settings for Dictionary.....	54
21. SUBLIME TEXT – PACKAGES.....	56
22. SUBLIME TEXT – MENUS	59
Save with Encoding	59
Convert Case	60

Goto Symbol	61
23. SUBLIME TEXT – SUB MENUS OF FONT	62
24. SUBLIME TEXT – DEVELOPING PLUGIN.....	64
Developing Plugin	64
Running the plugin	66
25. SUBLIME TEXT – COMMAND PALETTE.....	67
Using Command Palette	67
26. SUBLIME TEXT – DEBUGGING PHP APPLICATION	69
Installation of Sublime Plugin.....	69
27. SUBLIME TEXT – DEBUGGING JAVASCRIPT APPLICATION	72
Installation of Sublime Web Inspector	72
28. SUBLIME TEXT – BATCH PROCESSING.....	74
29. SUBLIME TEXT – DISTRACTION FREE MODE	76
Customization	77
30. SUBLIME TEXT – SUBLIMECODEINTEL PLUGIN	78
Installation	78
Configuration	80

1. Sublime Text – Introduction

Sublime Text Editor is a full featured Text editor for editing local files or a code base. It includes various features for editing code base which helps developers to keep track of changes. Various features that are supported by Sublime are as follows:

- Syntax Highlight
- Auto Indentation
- File Type Recognition
- Sidebar with files of mentioned directory
- Macros
- Plug-in and Packages

Sublime Text editor is used as an Integrated Development Editor (IDE) like Visual Studio code and NetBeans. The current version of Sublime Text editor is 3.0 and is compatible with various operating systems like Windows, Linux and MacOS.

Why Sublime Text?

When you use a suitable Text editor, you can enjoy its rich beneficial features. Sublime Text offers its users the following benefits:

- Ability to solve linker errors.
- Keeping track of all files and folders to work with.
- Connectivity with version control systems like Git, Mercurial.
- Problem solving capabilities.
- Keeping color combination for syntax combination.

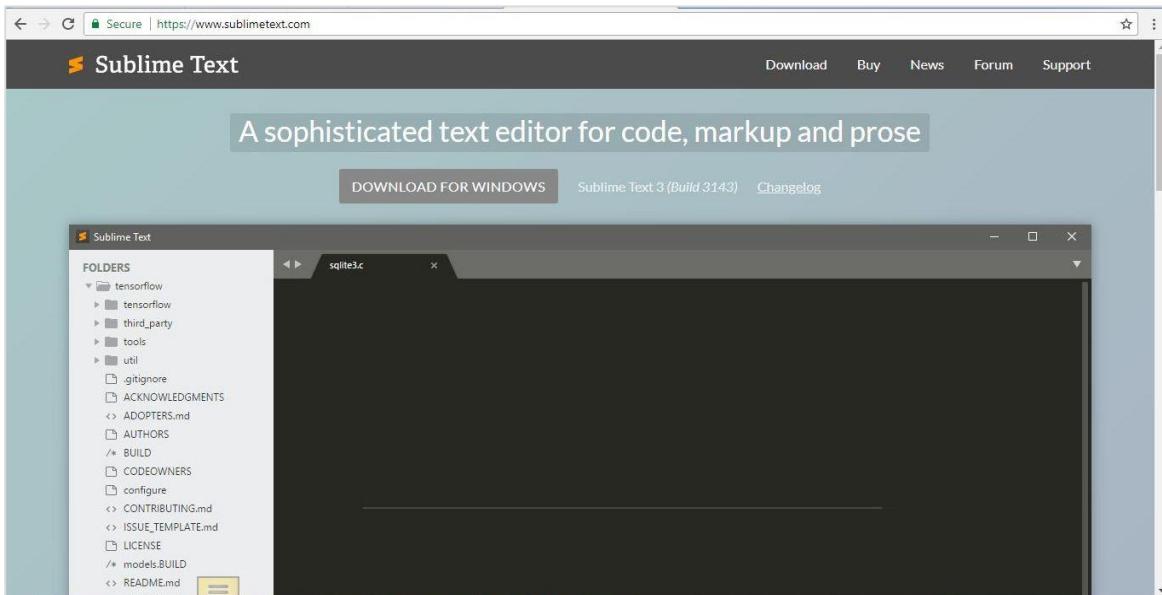
You can download Sublime Text from its official Website: www.sublimetext.com. In the next chapter, we will learn about installation procedure of Sublime Text editor in various operating systems.

2. Sublime Text – Installation

Sublime Text editor is supported by the following major operating systems:

- Windows
- Linux and its distributions
- OS X

You can download Sublime Text from its official website: www.sublimetext.com



In this chapter, you will learn about the installation of Sublime Text on various operating systems.

Installation on Windows

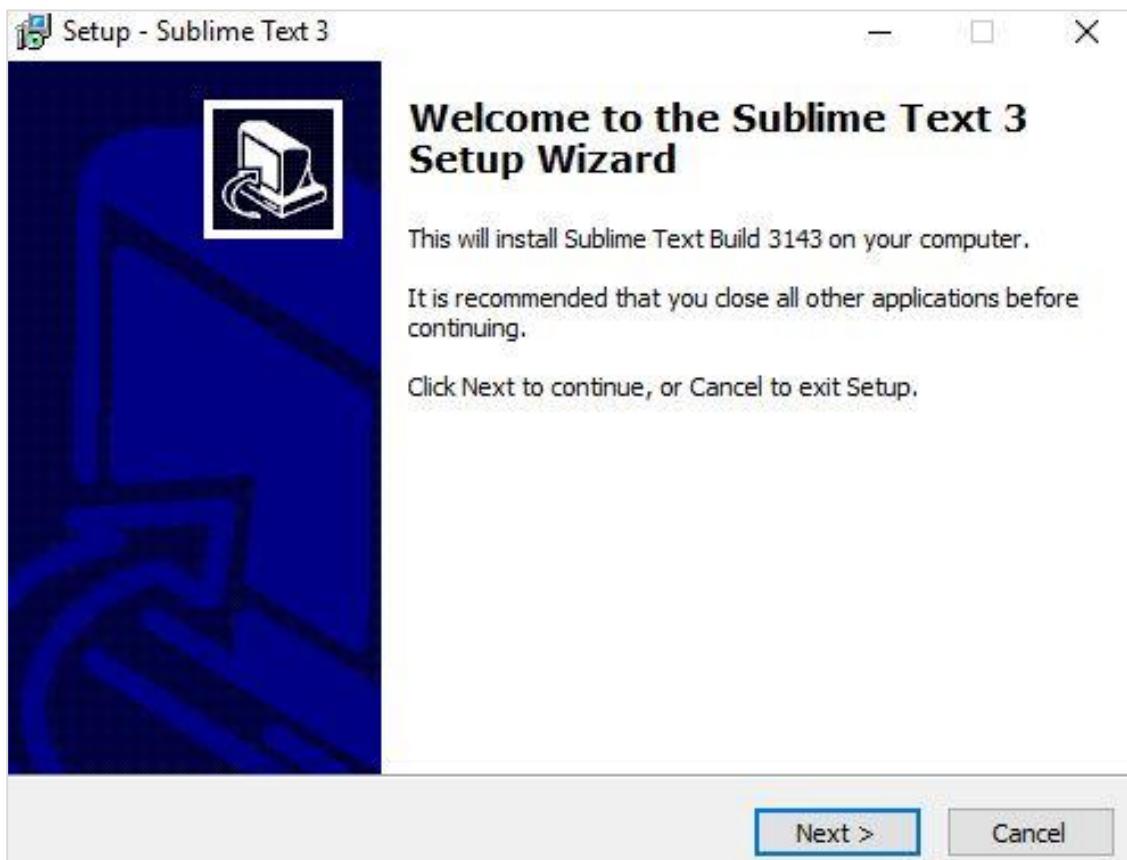
You will have to go follow the steps shown below to install Sublime Text on Windows:

Step 1: Download the **.exe** package from the official website as shown below:

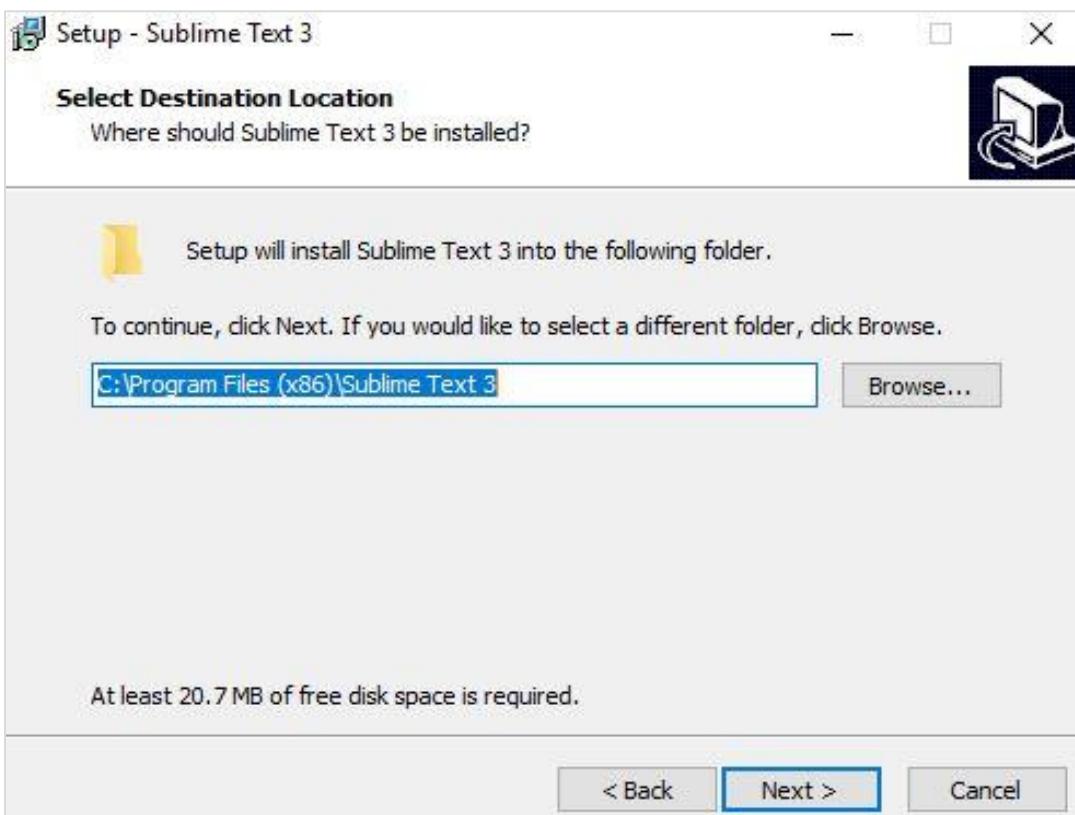
<https://www.sublimetext.com/3>

The screenshot shows the official Sublime Text website at <https://www.sublimetext.com/3>. The main navigation bar includes 'Download', 'Buy', 'News', 'Forum', and 'Support'. The 'Download' section is highlighted. Below it, there's a note about Sublime Text 3 being the current version. A list of download links for 'Version: Build 3143' is provided, including links for OSX, Windows, Windows 64-bit, and Linux repos. To the right, there's a sidebar with a note about following @sublimehq on Twitter and links for 'OTHER DOWNLOADS' like 'Dev Builds' and 'Sublime Text 2'.

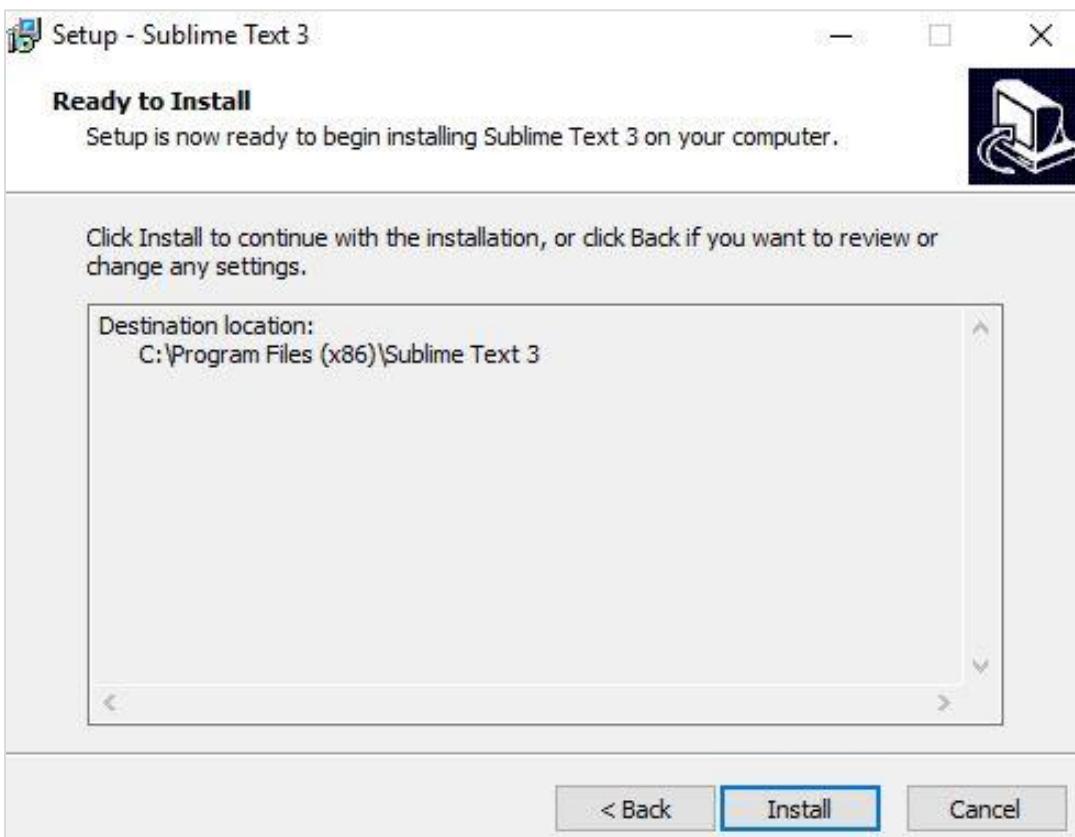
Step 2: Now, run the executable file. This defines the environment variables. When you run the executable file, you can observe the following window on your screen. Click **Next**.



Step 3: Now, choose a destination location to install Sublime Text3 and click **Next**.



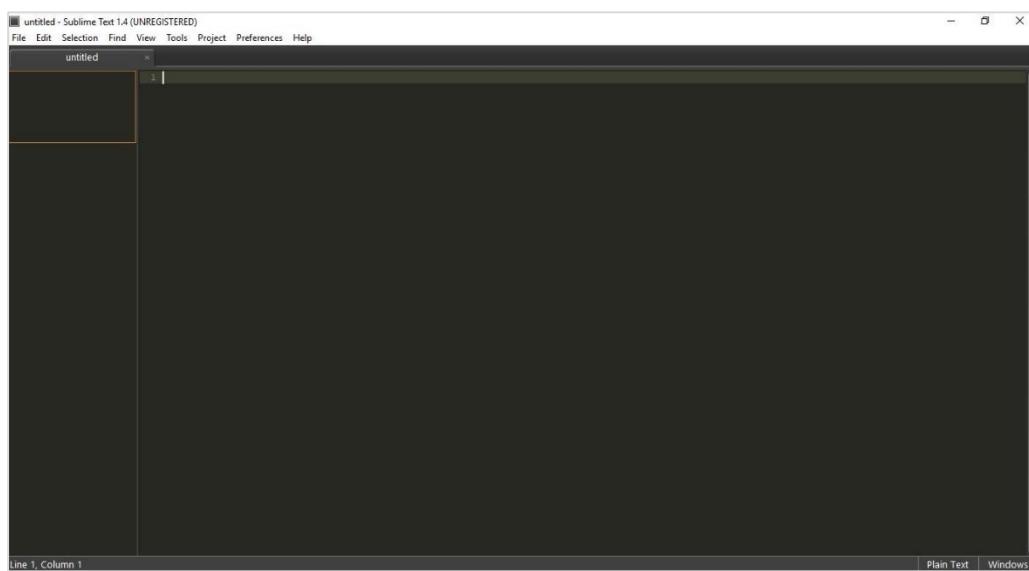
Step 4: Verify the destination folder and click **Install**.



Step 5: Now, click **Finish** to complete the installation.



Step 6: Upon a successful installation, your editor will appear as shown below:



Installation on Linux

You will have to follow the steps shown below to install Sublime Text on Linux distributions:

Step 1: Using the command line terminal, install the packages for Sublime Text editor, using the command given below:

```
sudo add-apt-repository ppa:webupd8team/Sublime-Text-3
```

Step 2: Update the packages using the following command:

```
sudo apt-get update
```

Step 3: Install the Sublime Text repository using the following command:

```
sudo apt-get install Sublime-Text
```

After the successful execution of above mentioned commands, you will find that Sublime Text editor is installed on the system.

Installation on OSX

For OSX operating systems,

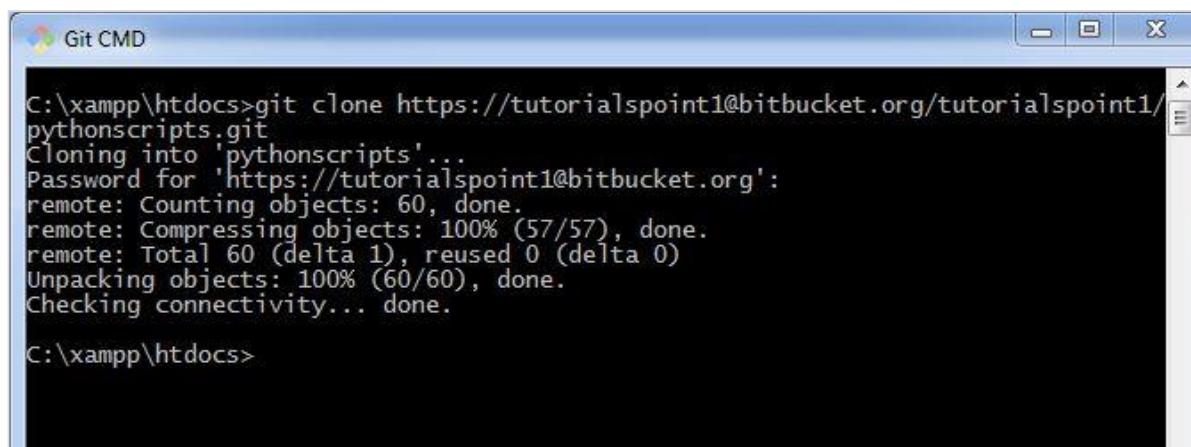
- Download the **.dmg** file of Sublime Text Editor.
- Open it and drag-and-drop in the **Applications** folder.
- Follow the steps that you have seen in above two cases.
- Launch the application.

3. Sublime Text – Data Directory

Throughout this tutorial, we will be focusing on using the subversion control system, Git and bit bucket in combination with Sublime Text editor.

As with any other Text editor, working on a specific repository is the key aspect of Sublime Text. It is shown in detail in the steps given below:

Step 1: Consider a repository to be cloned from the bit bucket. Please note that we will be working with Python scripts throughout this tutorial.

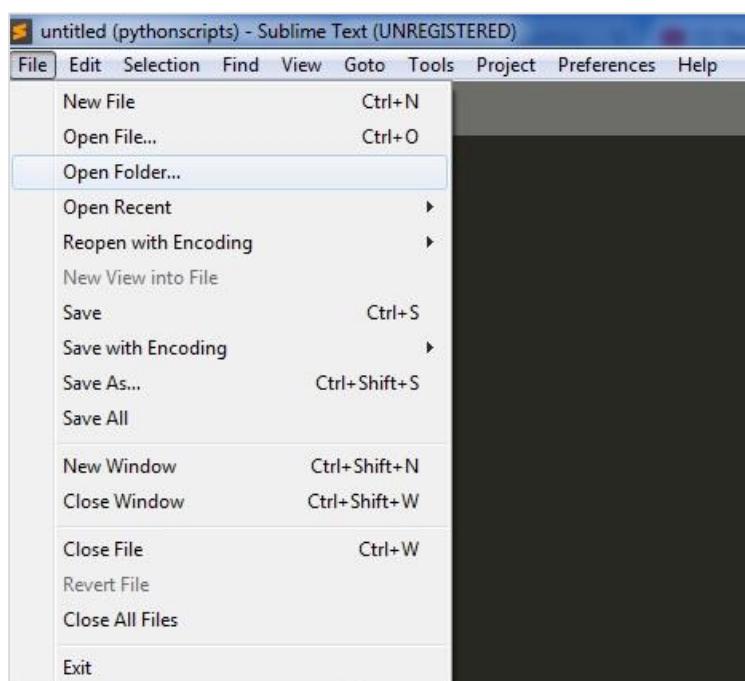


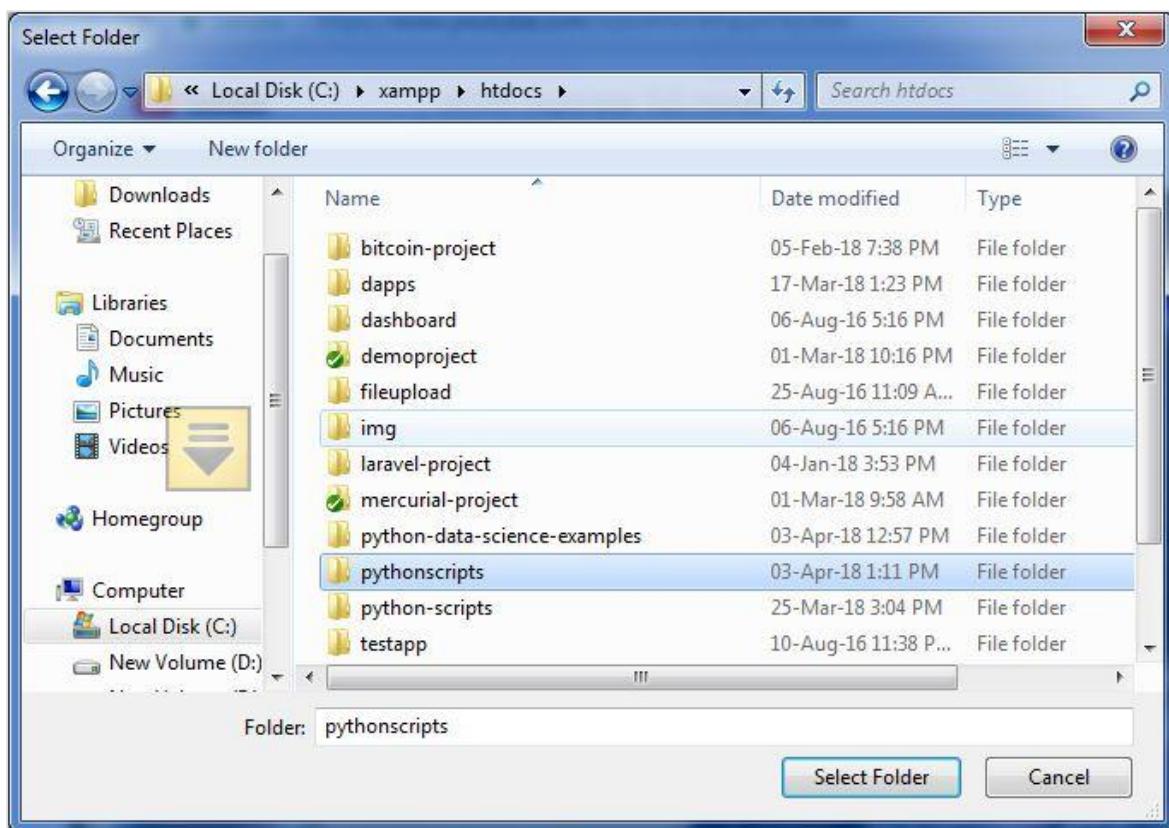
```
Git CMD

C:\xampp\htdocs>git clone https://tutorialspoint1@bitbucket.org/tutorialspoint1/pythonscripts.git
Cloning into 'pythonscripts'...
Password for 'https://tutorialspoint1@bitbucket.org':
remote: Counting objects: 60, done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 60 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (60/60), done.
Checking connectivity... done.

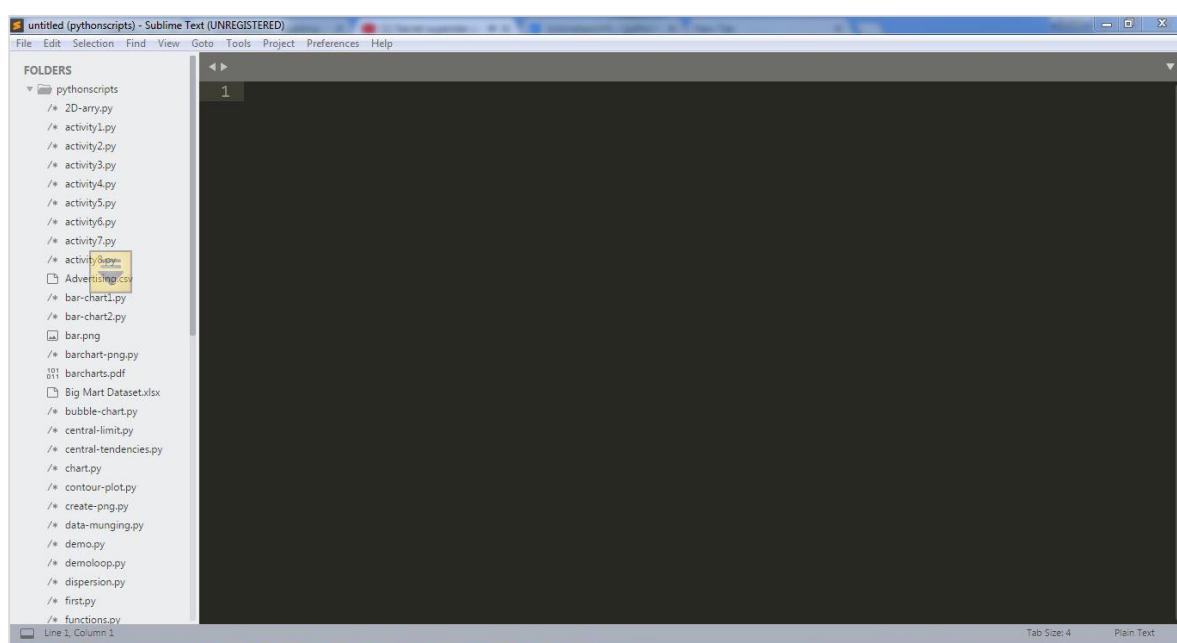
C:\xampp\htdocs>
```

Step 2: Include the repository in Sublime Text editor using the **Open Folder** option, as shown below.

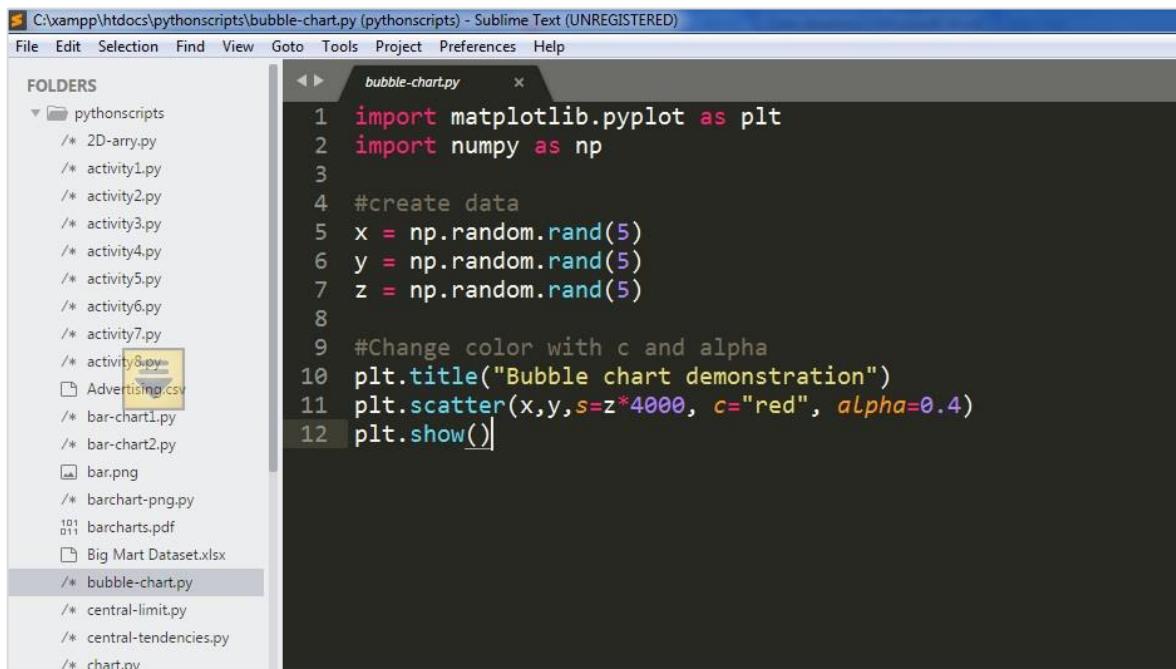




Step 3: After including the repository, the screen of Sublime Text editor will be as shown below:



You can choose the file you wish to open from the list that is displayed in the left hand side of the screen, as shown in the image below:



The screenshot shows the Sublime Text interface. The title bar reads "C:\xampp\htdocs\pythonscripts\bubble-chart.py (pythonscripts) - Sublime Text (UNREGISTERED)". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. On the left, there's a sidebar titled "FOLDERS" showing a directory structure with files like 2D-arry.py, activity1.py, etc., and a CSV file named Advertising.csv. The main editor window displays Python code for creating a bubble chart:

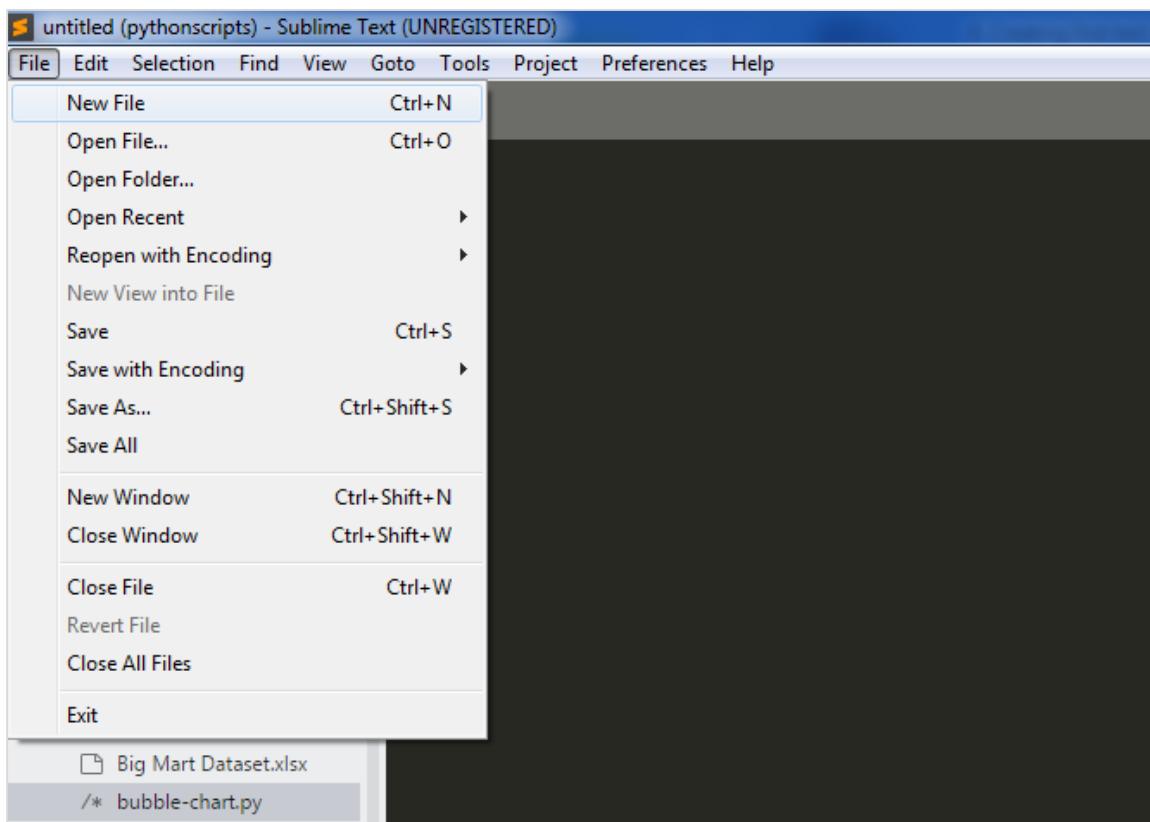
```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 #create data
5 x = np.random.rand(5)
6 y = np.random.rand(5)
7 z = np.random.rand(5)
8
9 #Change color with c and alpha
10 plt.title("Bubble chart demonstration")
11 plt.scatter(x,y,s=z*4000, c="red", alpha=0.4)
12 plt.show()
```

Note that the color combination gives recognition of keywords and packages included in the code.

4. Sublime Text – Creating First Document

In this chapter, you will learn step by step how to create a file in Sublime Text.

Step 1: Use the option **New File** to create a file in Sublime Text editor.



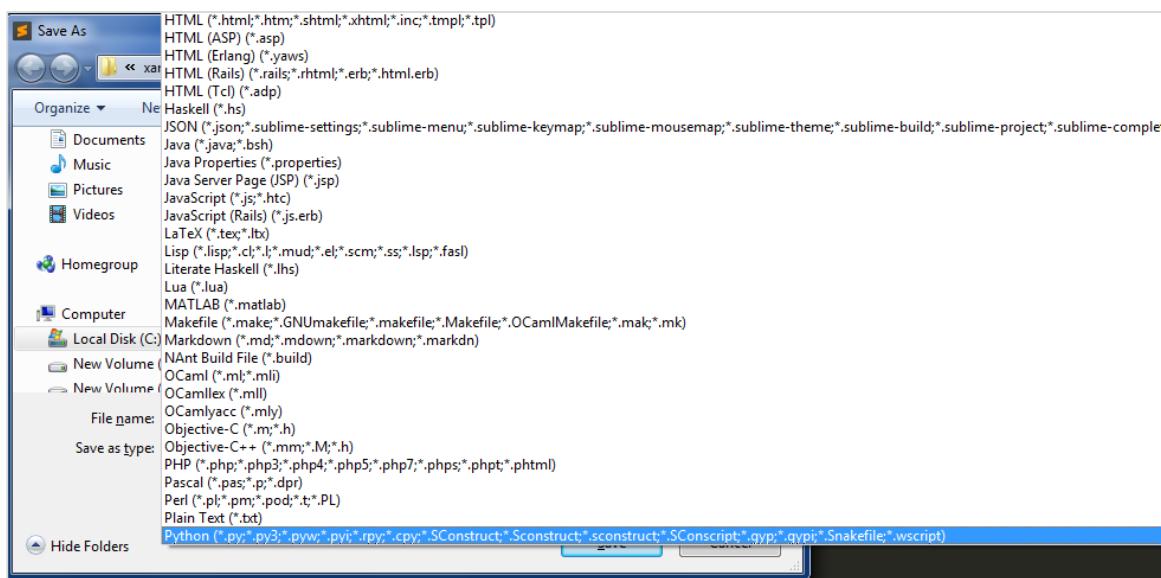
Step 2: Once the new file is available, you can insert the code, similar to any other text editor, and save the file in your working repository.

```
C:\xampp\htdocs\pythonscripts\demo1.py (pythonscripts) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

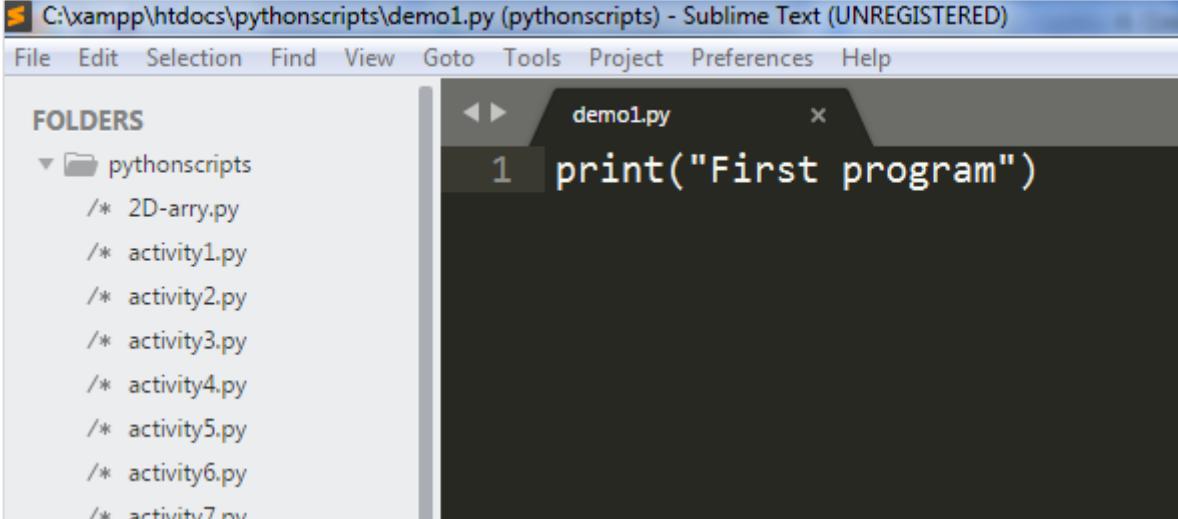
FOLDERS
pythonscripts
/* 2D-arry.py
/* activity1.py
/* activity2.py
/* activity3.py
/* activity4.py
/* activity5.py
/* activity6.py
/* activity7.py

demo1.py
1 print("First program")
```

Step 3: Once you save the associated file, Sublime Text editor provides a variety of options based on the type of file, for example **.php** for PHP scripts, **.py** for Python scripts and **.java** for Java code base.



Step 4: As our code repository is on Python project, we will save the file as **demo1.py**, as shown below:



The screenshot shows the Sublime Text interface. The title bar indicates the file is 'C:\xampp\htdocs\pythonscripts\demo1.py (pythonscripts) - Sublime Text (UNREGISTERED)'. The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. On the left, there's a sidebar titled 'FOLDERS' showing a 'pythonscripts' folder containing several Python files: 2D-arry.py, activity1.py, activity2.py, activity3.py, activity4.py, activity5.py, activity6.py, and activity7.py. The main editor window is titled 'demo1.py' and contains the single line of code: 'print("First program")'.

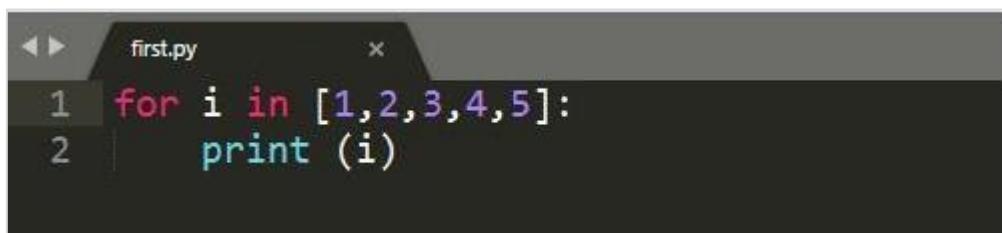
5. Sublime Text – Editing First Text Document

Code editors allow the users to edit the code scripts and text documents with various shortcut keys. In this chapter, let us understand through various examples about editing options of first text document in Sublime Text.

Simultaneous Write-Up

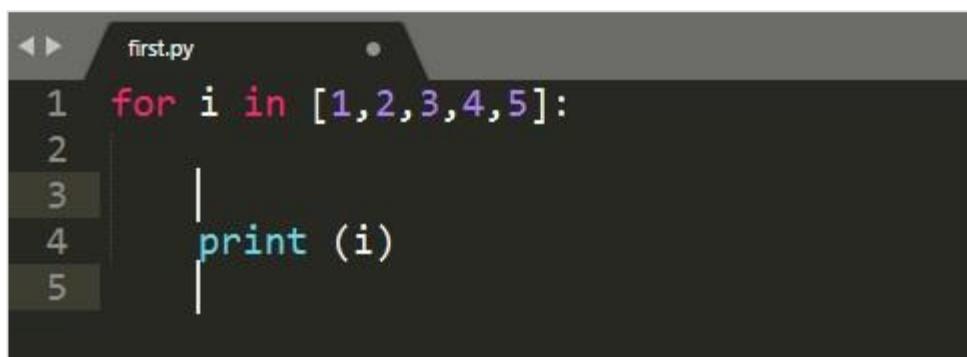
In this example, you will see how to add print statements at two locations.

Step1: Consider the first script of Python with the following code in it:



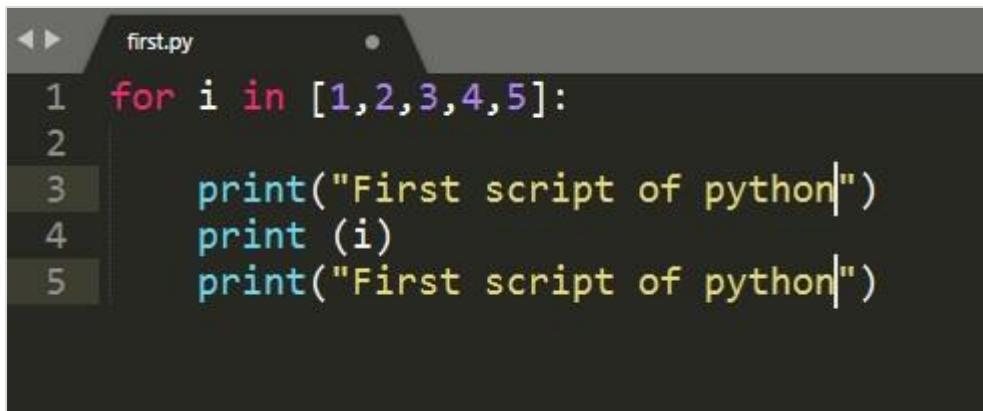
```
first.py
1 for i in [1,2,3,4,5]:
2     print (i)
```

Step 2: Let us suppose that, in the given script you want to mention the starting and ending points of the loop, using simple statements. You can write the print statement wherever required, however, in Sublime Text editor you can simultaneously add comments and statements in two places with a shortcut of **Ctrl+cursor** point for Windows or Linux, and **Cmd+cursor** point for Mac. Then, you can see the cursor points as mentioned below:



```
first.py
1 for i in [1,2,3,4,5]:
2
3     | print (i)
4
5
```

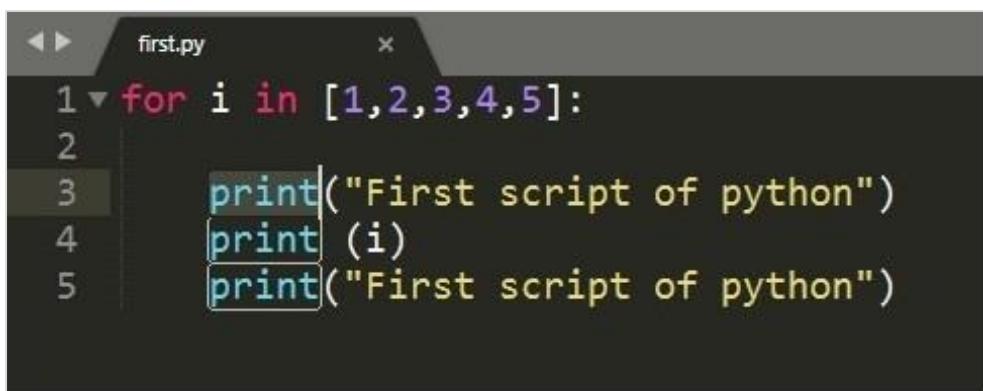
Step 3: Now, you can insert the print statements at both the locations of the mentioned cursor points, as shown in the image below.



```
first.py
1 for i in [1,2,3,4,5]:
2
3     print("First script of python")
4     print (i)
5     print("First script of python")
```

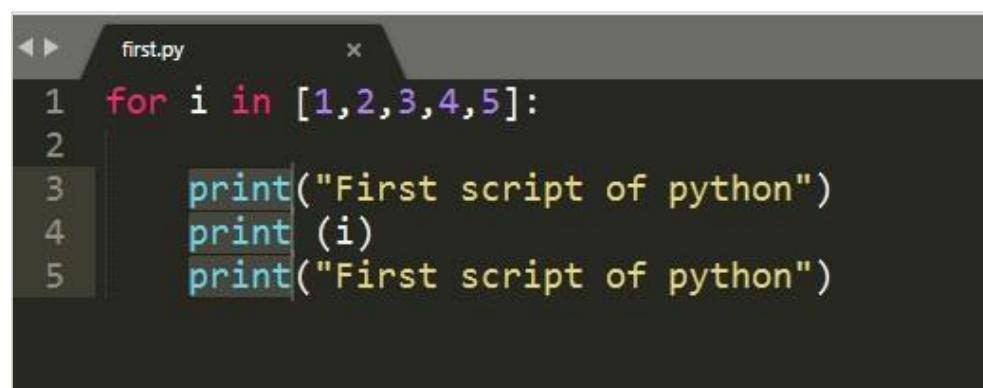
Finding Occurrences

Sublime Text editor includes a feature to find the occurrences of the keywords included in the scripts. The shortcut key for finding occurrences of the keyword is **Ctrl+D** after highlighting the associated keyword.



```
first.py
1 for i in [1,2,3,4,5]:
2
3     print("First script of python")
4     print (i)
5     print("First script of python")
```

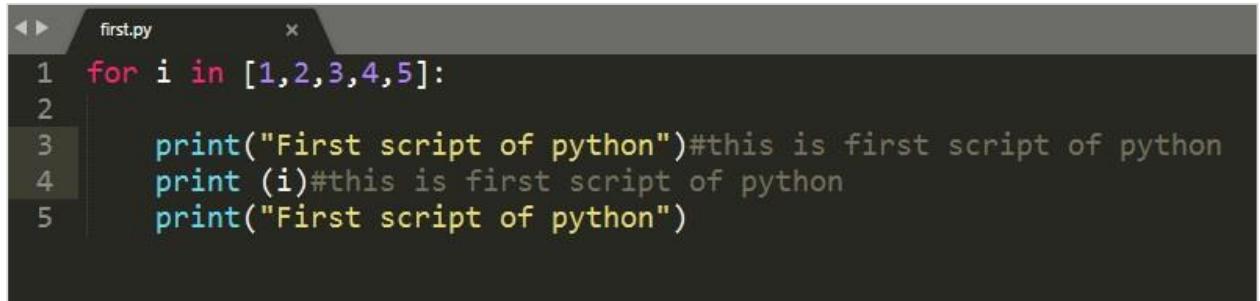
If you want to search for a keyword, say **print** from the given code, you can use **Ctrl+D** or **Cmd+D** to get the count of occurrences of the associated keyword.



```
first.py
1 for i in [1,2,3,4,5]:
2
3     print("First script of python")
4     print (i)
5     print("First script of python")
```

Appending Comments in Line

We can append comments at the end of line using the shortcut key **Ctrl+Shift+L** for Windows and **Cmd+Shift+L** for Mac operating system after selecting the code section where you actually need the comment.



```
first.py
1 for i in [1,2,3,4,5]:
2
3     print("First script of python")#this is first script of python
4     print (i)#this is first script of python
5     print("First script of python")
```

6. Sublime Text – Patterns of Code Editing

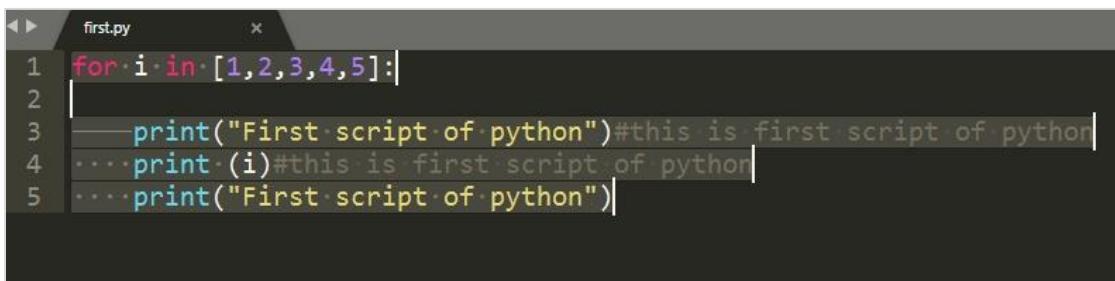
There are various types of code editing and shortcut keys used in Sublime Text editor:

- Splitting the selection into lines
- Wrapping the complete paragraph in HTML file
- Finding all occurrences

This chapter discusses each of them in detail.

Splitting the Selection into Lines

The key combination **Ctrl+Shift+L** allows you to select the blocks of lines and split them, on Windows. The screenshot given below explains this:



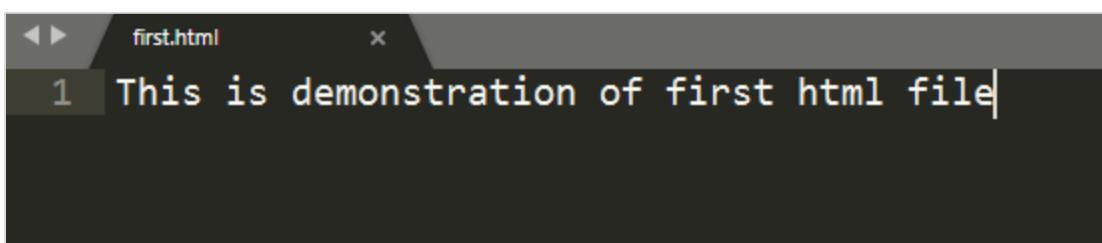
A screenshot of the Sublime Text editor showing a Python script named 'first.py'. The code contains a for loop that prints two lines of text. The second line of the print statement is highlighted with a yellow selection bar, indicating it is selected for splitting.

```
1 for i in [1,2,3,4,5]:  
2 |  
3 |     print("First script of python")#this is first script of python  
4 |     print(i)#this is first script of python  
5 |     print("First script of python")|
```

Wrapping Complete Paragraph in HTML File

The key combination **Alt+Shift+w** for Windows and **Cmd+Shift+w** for Mac, allows the user to split the sections of code and create various paragraphs. It is referred as a **wrap selection with html tag**. This also allows you to include HTML tags using Sublime editor.

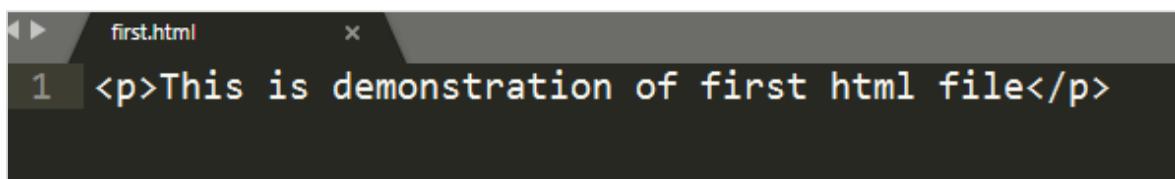
The screenshot given below explains wrapping text with a HTML tag:



A screenshot of the Sublime Text editor showing an HTML file named 'first.html'. The text 'This is demonstration of first html file' is selected and wrapped in a single <p> tag.

```
1 This is demonstration of first html file|
```

After wrapping the text, the text wrapped with a HTML tag will be seen as shown below:



A screenshot of the Sublime Text editor showing the same HTML file 'first.html'. The text is now displayed within a single <p> tag.

```
1 <p>This is demonstration of first html file</p>
```

Finding all Occurrences

The key pattern **Ctrl+Shift+f** allows you to find a keyword in a given project. The screenshot given below explains this:

The screenshot shows the Sublime Text interface with a Python script named 'first.py' open. The script contains the following code:

```

1 for i in [1,2,3,4,5]:
2
3     print("First script of python")#this is first script of python
4     print (i)#this is first script of python
5     print("First script of python")

```

The file tree on the left shows various Python files and other files like 'Advertising.csv', 'bar-chart1.py', etc. The 'first.py' file is selected.

In the bottom right corner, there is a 'Find' interface with the following fields:

- Find: (empty)
- Where: C:\xampp\htdocs\pythonscripts
- Replace: (empty)

The 'Where:' field is highlighted. The 'Find' button is also highlighted.

The shortcut key opens a window with three options included in it, namely **find**, **where** and **replace**,

where,

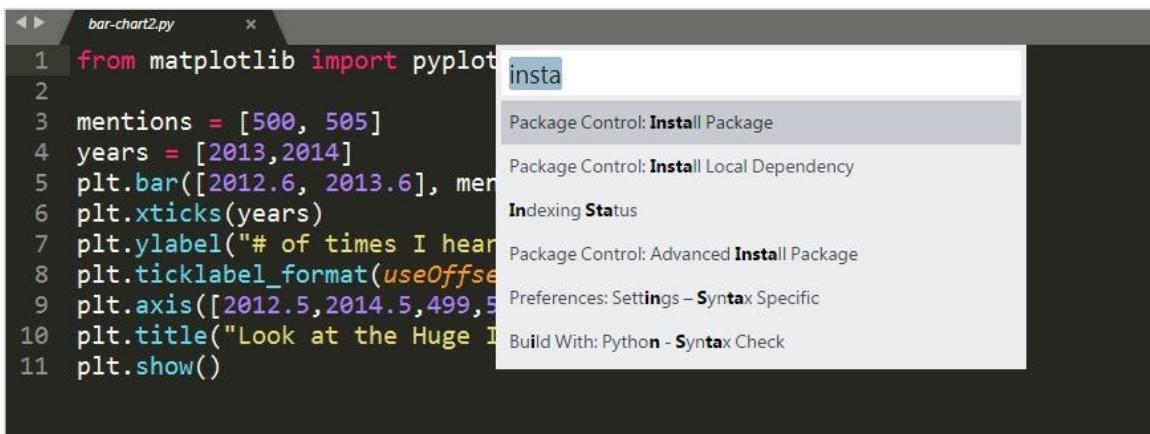
- **find** refers to the search of the particular keyword,
- **where** mentions the section as to which repository is searched, and
- **replace** mentions the keyword which is replaced instead of find keyword.

7. Sublime Text – Sublime Linter

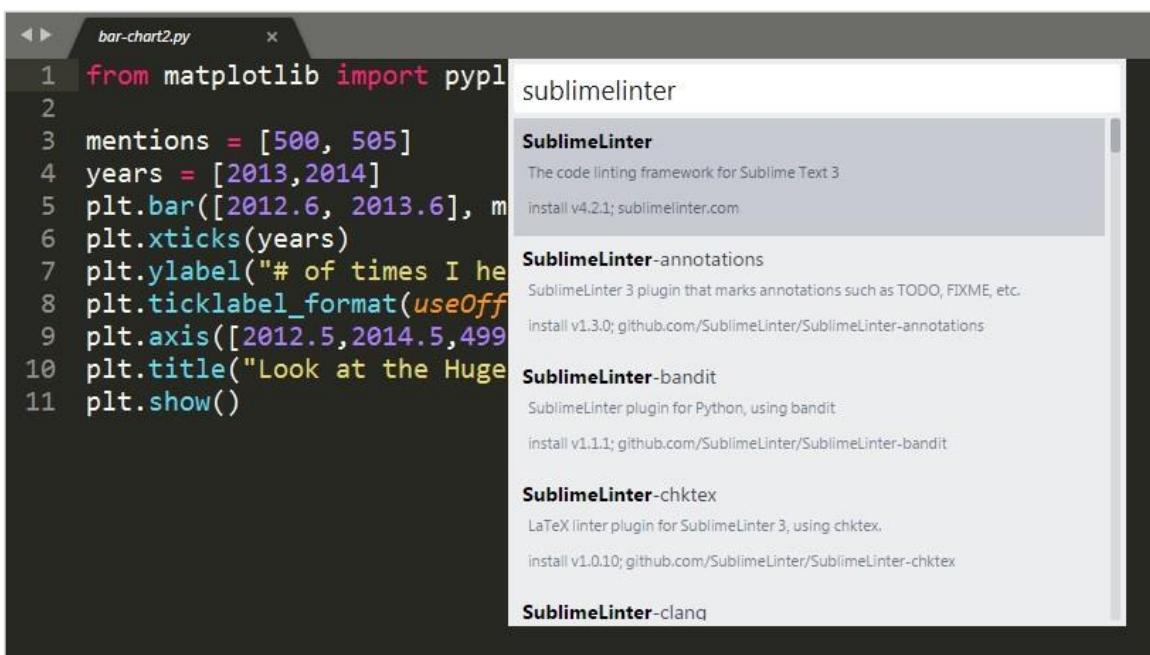
Linting is the process of flagging suspicious constructs and likely to be bugs in any programming language.

Sublime Linter is the package of Sublime Text that helps you to perform linting in Sublime. It helps in creating files in different modes for the manipulation of code base. It is not an in built package and you need to install it manually.

You can install any package control in Sublime Text editor using the shortcut key combination **Ctrl+Shift+P**, and selecting the **Package Control: Install Package** option.



Select the associated package to install in Sublime Text editor. To install Sublime Linter, you need to select the option of **SublimeLinter** plugin.



On successful installation, your screen will look as shown below:

The screenshot shows the Sublime Text interface with the title bar "Package Control Messages (pythonscripts) - Sublime Text (UNREGISTERED)". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The left sidebar shows a file tree with a folder "pythonscripts" containing various Python files like 2D-arry.py, activity1.py, etc., and some CSV and XLSX files. The main view displays the "bar-chart2.py" file. The status bar at the bottom says "SublimeLinter has been installed or upgraded. Please restart Sublime Text." A yellow icon with a downward arrow is also visible.

```

1 Package Control Messages
2 =====
3
4 SublimeLinter
5 -----
6
7
8
9
10
11
12
13
14
15 Welcome to SublimeLinter, a linter framework for Sublime Text 3.
16 Linters are not included, they must be installed separately.
17
18
19 For complete documentation on how to use and configure SublimeLinter,
20 please see: http://www.sublimelinter.com
21

```

Modes of Sublime Linter

Sublime Linter runs in four different modes as mentioned below:

Background Mode

When the Sublime Linter key is set to **true**, linting is performed in the background constantly each time the file is modified.

Load-Save Mode

When the Sublime Linter key is set to **load save**, linting will be performed when a file is loaded from the repository.

Save-Only Mode

When the Sublime Linter key is set to **save only** mode, linting is performed while saving the particular file in working repository.

On Demand Mode

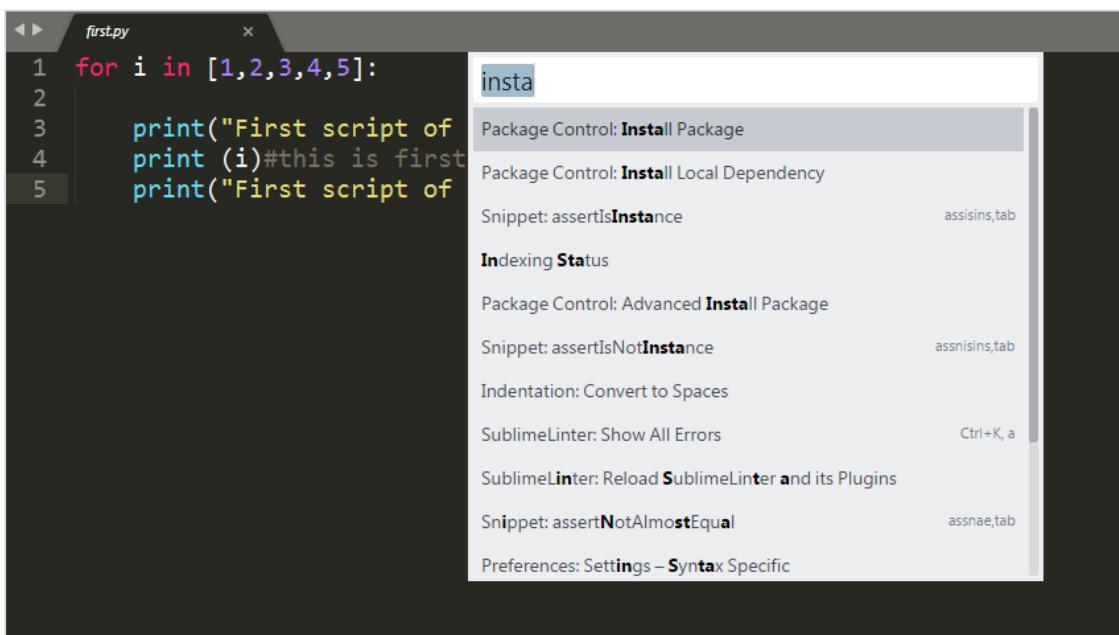
In the on-demand mode, the Sublime Linter will be set to **true**, using the shortcut **Ctrl+Alt+L**, which sets up the linter to detect bugs, if any present in the code.

8. Sublime Text – Shortcuts

Sublime Text editor includes shortcuts and methods for manipulating the output. This chapter discusses in detail about these shortcuts with the help of suitable illustrations.

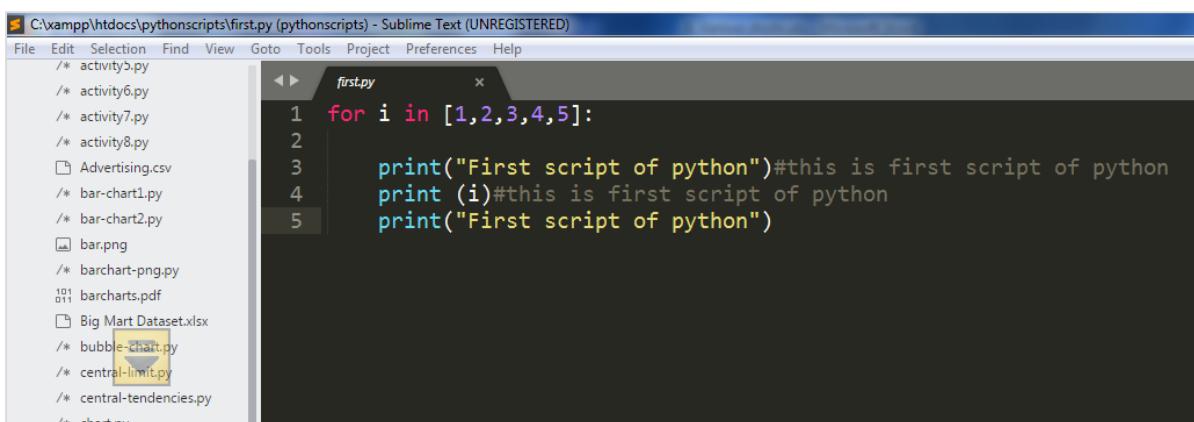
Opening the Command Palette

Command palette in Sublime Text editor includes options for installing packages and console. The shortcut key for this purpose is **Ctrl+Shift+P** for Windows and **Cmd+Shift+P** for Mac. The screenshot given below shows opening the command palette using this shortcut.



Toggle Side Bar

Sublime text editor includes a side bar which displays the scripts or the file names. You can use the shortcut key **Ctrl+KB** on Windows and **Cmd+KB** for Mac for this purpose.



C:\xampp\htdocs\pythonscripts\first.py (pythonscripts) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

first.py

```

1 for i in [1,2,3,4,5]:
2
3     print("First script of python")#this is first script of python
4     print (i)#this is first script of python
5     print("First script of python")

```

source.python

Display Scope in Status Bar

The shortcut key **Ctrl+Shift+Alt+P** for Windows and **Ctrl+Shift+P** for Mac displays scope in the status bar. The following screenshot shows an illustration of this shortcut.

first.py

```

1 for i in [1,2,3,4,5]:
2
3     print("First script of python")#this is first script of python
4     print (i)#this is first script of python
5     print("First script of python")

```

source.python
meta.function-call.python
punctuation.section.arguments.end.python

[Copy](#)

Python Console

When you are working with Python scripts using Sublime Text editor, you can use **Ctrl+`** for Windows or **Control + `** for Windows, to operate the Python console.

C:\xampp\htdocs\pythonscripts\first.py (pythonscripts) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

- pythonscripts
 - /* 2D-arry.py
 - /* activity1.py
 - /* activity2.py
 - /* activity3.py
 - /* activity4.py
 - /* activity5.py
 - /* activity6.py
 - /* activity7.py
 - /* activity8.py
 - /* Advertisings.csv
 - /* bar-chart1.py
 - /* bar-chart2.py
 - bar.png
 - barchart-png.py
 - barcharts.pdf
 - Big Mart Dataset.xlsx
 - bubble-chart.py
 - central-limit.py
 - central-tendencies.py

```

1 for i in [1,2,3,4,5]:
2
3     print("First script of python")#this is first script of python
4     print (i)#this is first script of python
5     print("First script of python")

```

reloading plugin SublimeLinter.panel_view
reloading plugin SublimeLinter.status_bar_view
reloading plugin SublimeLinter.sublime_linter
plugins loaded
Package Control: Skipping automatic upgrade, last run at 2018-04-10 10:01:15, next run at 2018-04-10 11:01:15 or after

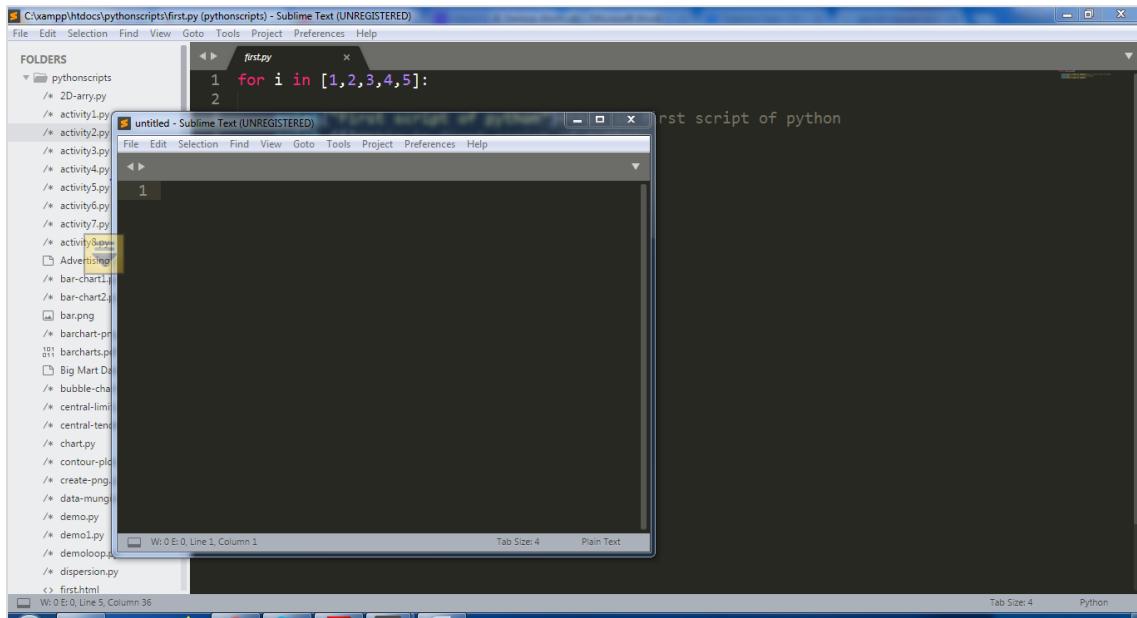
W:0 E:0, Line 5, Column 36

Tab Size: 4 Python

11:28 AM 10-Apr-18

New Window

With new windows, you can manage a number of projects simultaneously. You can use the shortcut key **Ctrl+Shift+N** on Windows and **Cmd+Shift+N** for Mac to create a new window on Sublime Text editor.

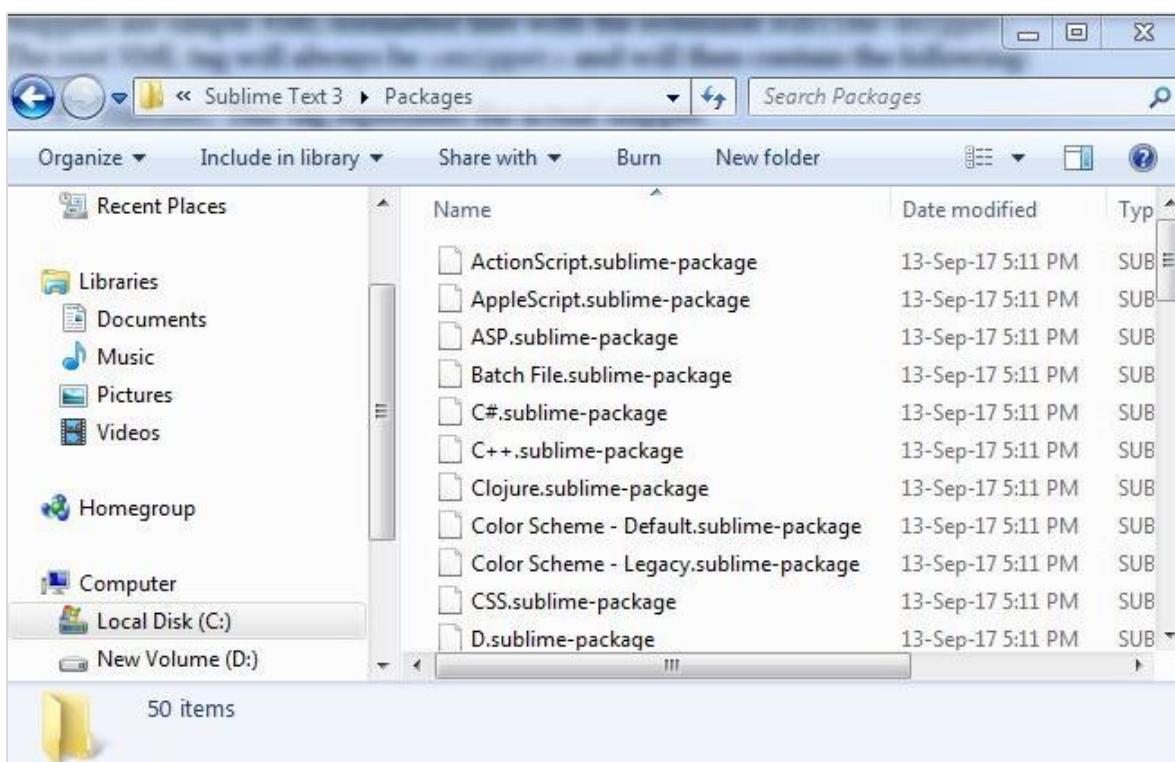


9. Sublime Text – Snippets

Snippets are the smart templates which can be reused whenever needed. Sublime text editor includes snippets feature for HTML templates. In this chapter, you will learn about them in detail.

Understanding Snippets

Snippets are simple XML supported files with various attributes. They include an extension called **sublime-snippet**. The root tag of XML element will be the **<snippet>** tag. Snippets created are saved in the **Packages** folder and are considered to live in this folder.

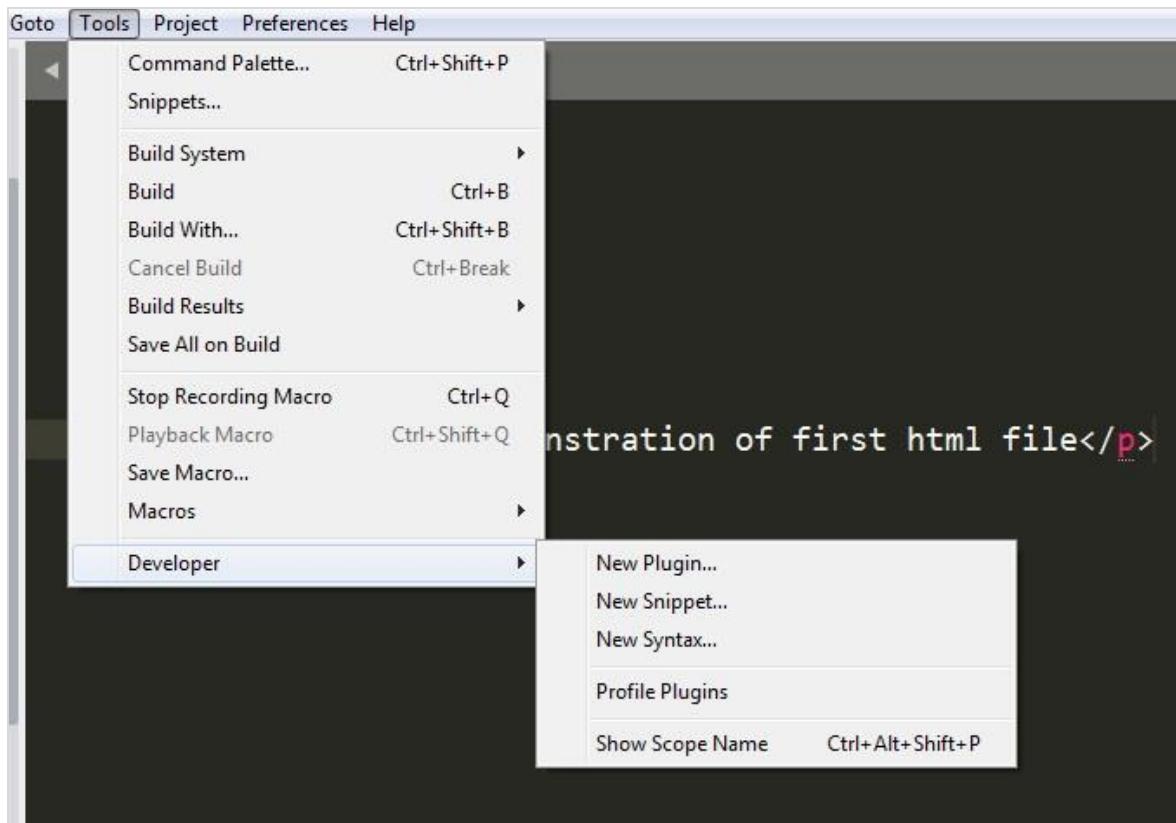


The file format and syntax of snippets are discussed in detail as follows:

- **Content:** This section includes description of the snippet.
- **tabTrigger:** It includes a sequence of characters which trigger when the snippet is written.
- **Scope:** It defines the scope in which the snippet remains active.
- **Description:** It includes all the meta description. It will be displayed when snippet's menu is open.

Creating First Snippet

You can create the default snippet using **Tools->Developer->Snippet** option.



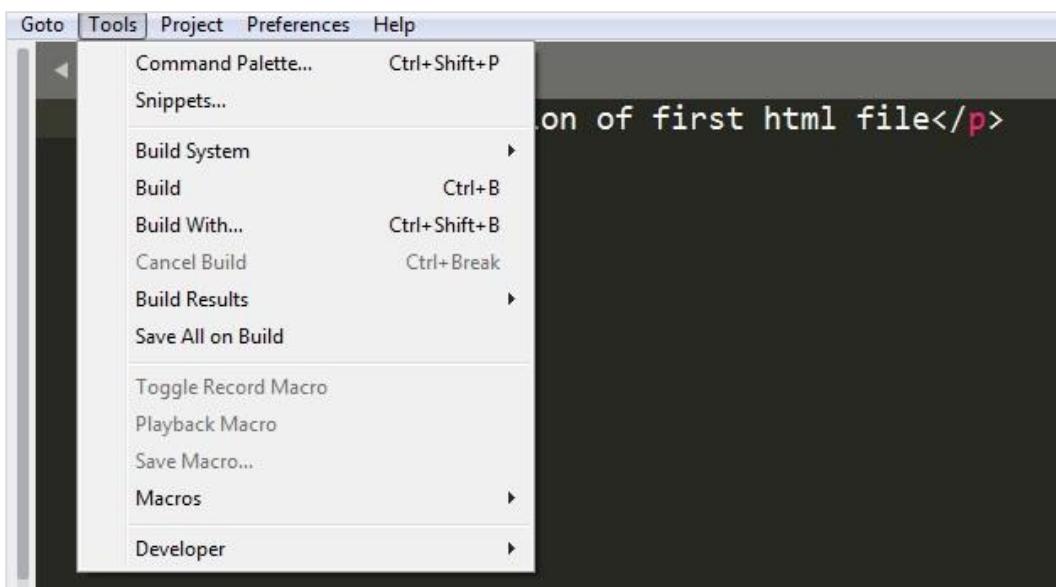
Then, a demo snippet with the associated tags will be created as shown below:

```

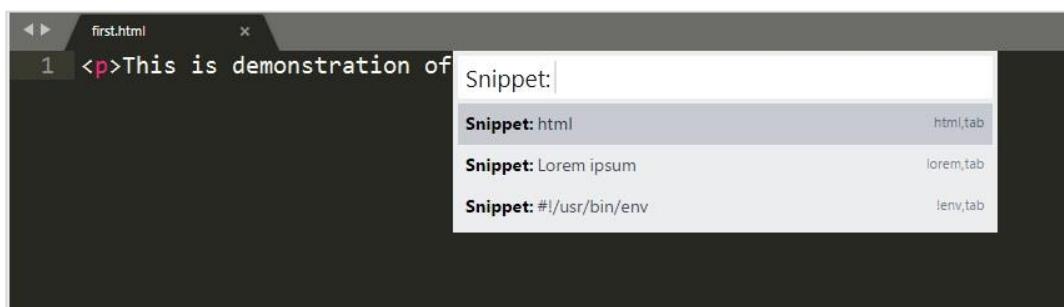
1 <snippet>
2   <content><![CDATA[
3 Hello, ${1:this} is a ${2:snippet}.
4 ]]></content>
5   <!-- Optional: Set a tabTrigger to define how to trigger the snippet -->
6   <!-- <tabTrigger>hello</tabTrigger> -->
7   <!-- Optional: Set a scope to limit where the snippet will trigger -->
8   <!-- <scope>source.python</scope> -->
9 </snippet>
10

```

To create a first snippet in Sublime Text editor, click the **Tools** menu and select the **Snippets** option from the drop down window, as shown in the screenshot here.



Now, choose **Snippet:html** from the options displayed.



This creates a default snippet on the specified html file as shown below.

A screenshot of the Sublime Text editor showing the 'first.html' file. The code is: 1 <!DOCTYPE html>, 2 <html>, 3 <head>, 4 <title>|</title>, 5 </head>, 6 <body>, 7 , 8 </body>, 9 </html>. The placeholder text '|</title>' is visible where the title was inserted. To the right of the code, there is a dark text editor window containing the code: <on of first html file</p>

Note that there are various snippets for different types of files. For html files in the code base, there are three snippets defined in Sublime Text editor, as shown below.

The screenshot shows a Sublime Text window with a file named "first.html". The code in the file is:

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title></title>
5 </head>
6 <body>
7
8 </body>
9 </html><p>This is demonstration of first html file</p>

```

A snippet menu is open on the right, with "Snippet:" typed into the search bar. The results show three snippets:

- Snippet: html** (html,tab)
- Snippet: Lorem ipsum** (lorem,tab)
- Snippet: #!/usr/bin/env** (env,tab)

Package Control Snippets

Control snippets are primarily used for product development. With **install packages** option in Sublime editor, you can download any snippet needed for web development and install it.

The screenshot shows a Sublime Text window with a file named "first.html". The code is the same as in the previous screenshot. A package search interface is open on the right, with "sni" typed into the search bar. The results show several packages related to snippets:

- Snippet Destroyer**: Destroy all Sublime Text completions and snippets. Version v1.0.2, installed from github.com/twolffson/sublime-snippet-destroyer.
- SnippetCaller**: Snippet system on top of default sublime snippets. Version v2016.05.13.05.21.28, installed from github.com/shagabutdinov/sublime-snippet-caller.
- SnippetMaker**: Makes managing snippets easy in Sublime Text. Version v2.1.0, installed from github.com/jugyo/SublimeSnippetMaker.
- SnippetManager**: Create snippet with hotkey. Version v2015.11.24.03.56.33, installed from github.com/shagabutdinov/sublime-snippet-manager.
- SnippetX**

You may need the following packages for web development:

- EverCodeLab Sublime supports for Ruby on Rails
- Additional PHP snippets for PHP
- HTML snippets for HTML files
- Twitter Bootstrap snippets for Twitter Bootstrap
- JavaScript snippets for JS files
- jQuery snippets pack for jQuery

10. Sublime – Macros

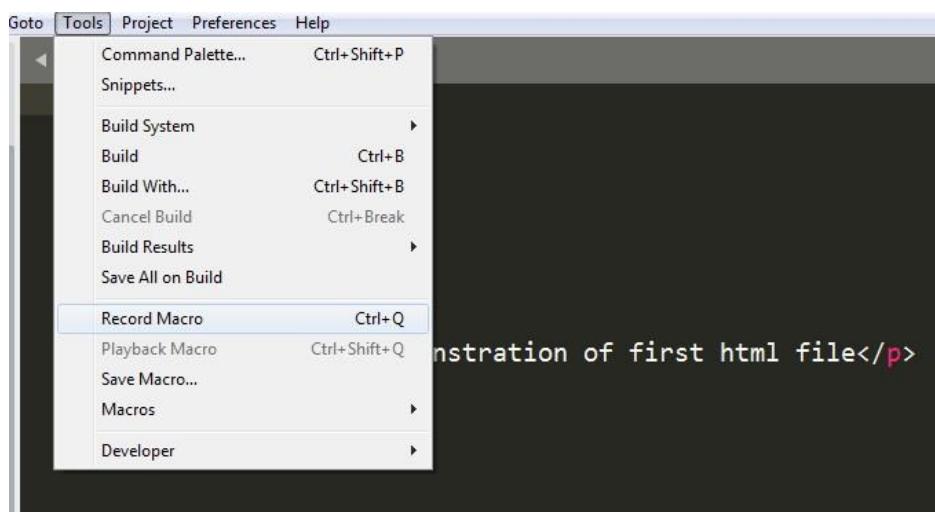
Macros are the basic automation facility that includes a sequence of commands. You can use macros whenever you need to perform an operation which includes same steps.

Macro files are JSON files with an extension called **.sublime-micro** and are considered to be very helpful.

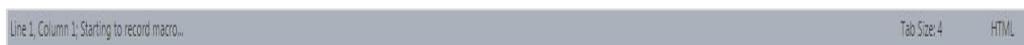
Recording a Macro

Recording a macro defines the procedure to keep a set of commands needed for the given file. You can record a macro in two ways:

- Use the shortcut key **Ctrl+Q** for recording a macro in Windows and **Cmd+Q** in Mac.
- Use the **record** option in **Tools -> Record Macro**.



Once the macro has started recording, you can see the following message in Sublime Text editor:

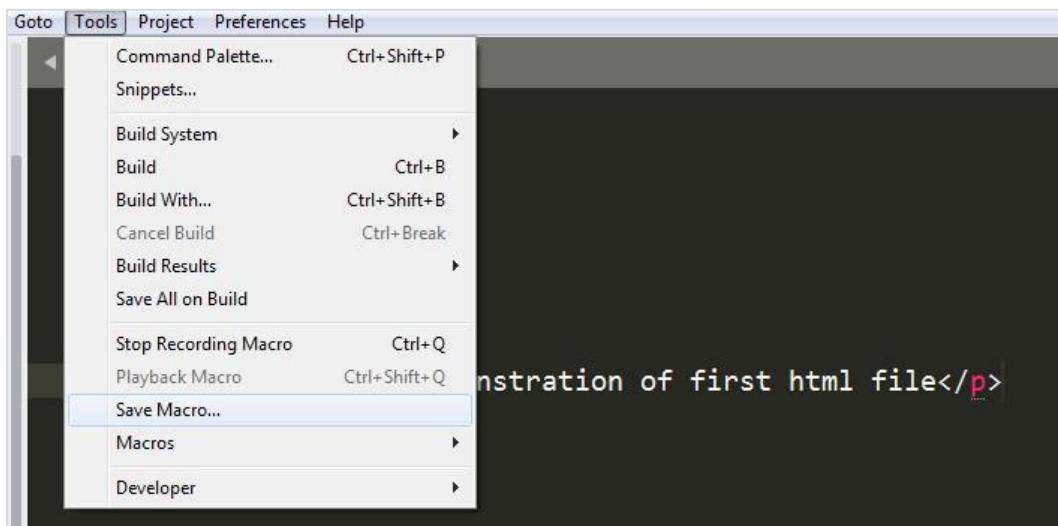


Playing a Macro

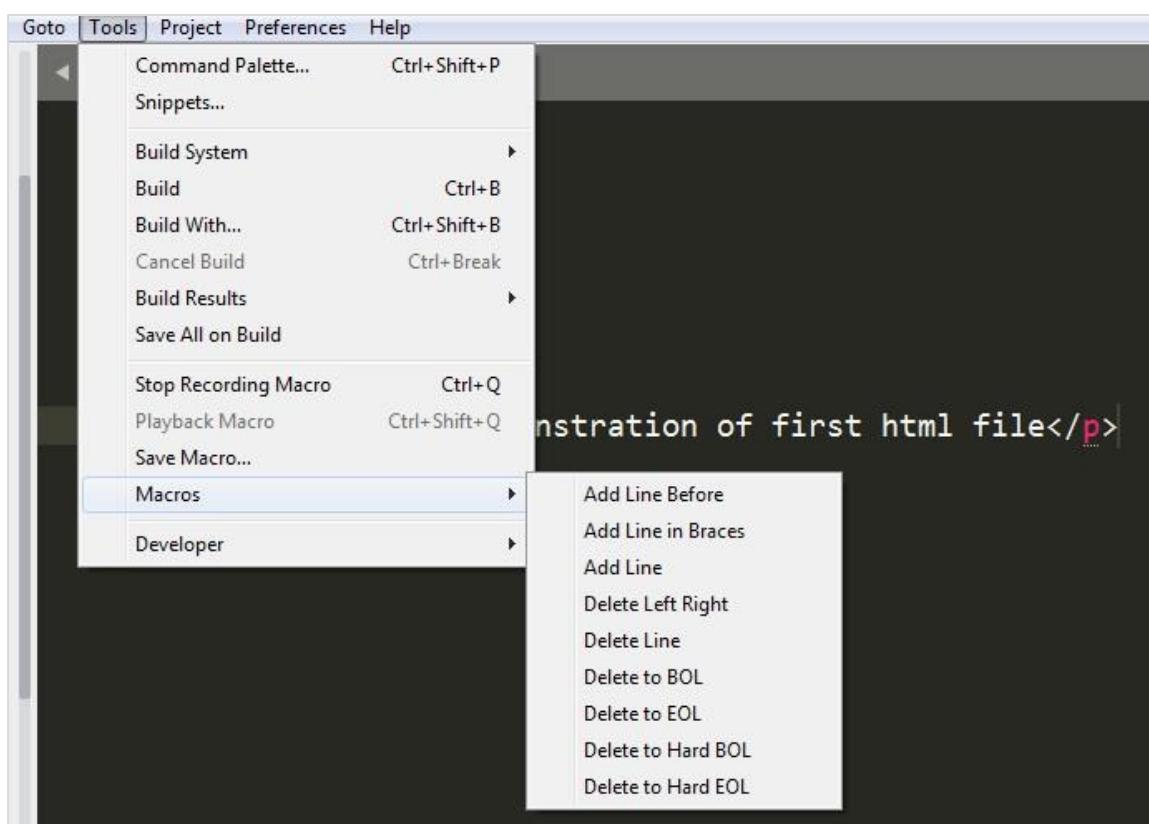
You can use the shortcut key **Ctrl+Shift+Q** on Windows and **Cmd+Shift+Q** for Mac, to play a macro recorded by the user in Sublime Text. Note that this shortcut plays the **last recorded** macro in Sublime.

Saving a Macro

You can save a recorded macro using the option **Tools -> Save Macro**. The following screenshot shows how to save a macro using this option.



You can also redefine the macros using various options in Tool menu bar, as shown below:

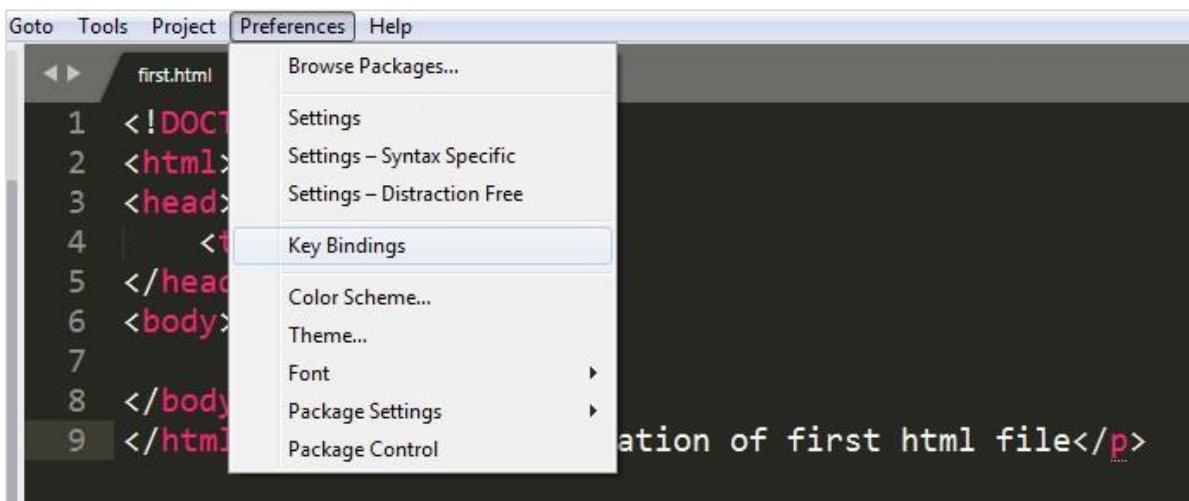


11. Sublime Text – Key Bindings

Key bindings in Sublime Text helps a user to process and map the sequences of key presses to actions. They are defined in the JSON format and are stored in **.sublime-keymap** files.

For better integration, it is important to keep separate key map files for Linux, OSX and Windows. Key maps of the corresponding platform will be loaded in the Sublime Text editor.

A user can open the **keymap** file or default key bindings using the option **Preferences -> Key Bindings**.

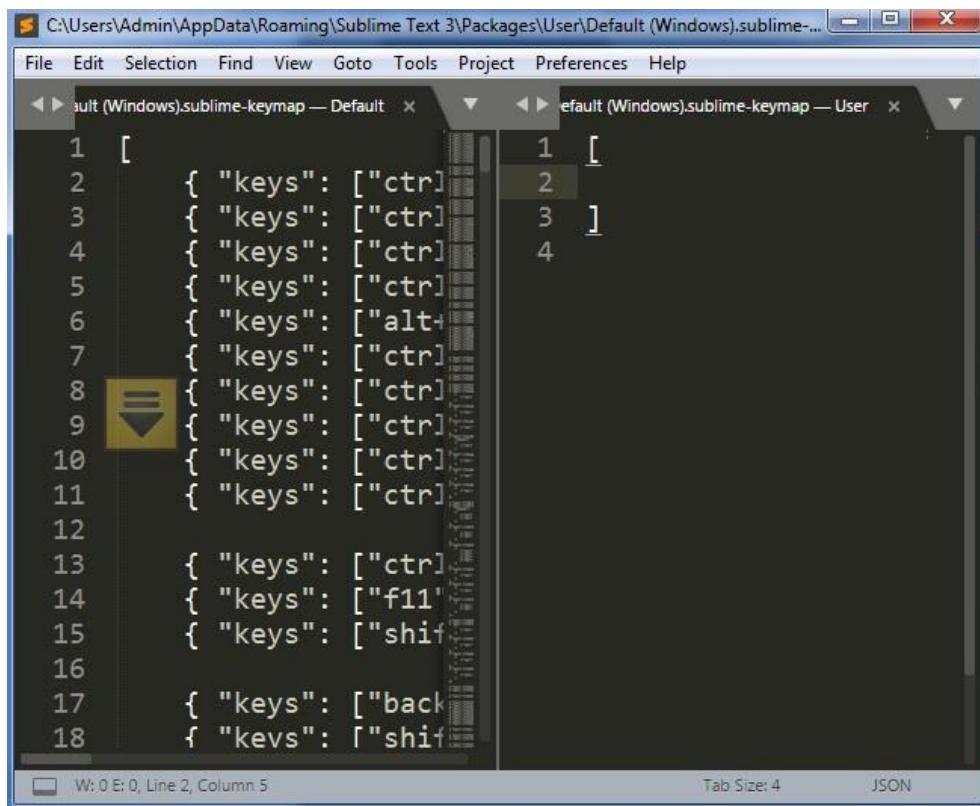
A screenshot of the Sublime Text interface showing two panels side-by-side. Both panels have the title "Default (Windows).sublime-keymap — Default" and "Default (Windows).sublime-keymap — User". The code in both panels is identical, displaying a JSON keymap definition. The code includes mappings for various keyboard shortcuts like Ctrl+Shift+N, Ctrl+Shift+W, Ctrl+O, Ctrl+Shift+T, Alt+O, Ctrl+N, Ctrl+S, Ctrl+Shift+S, Ctrl+F4, Ctrl+W, Ctrl+K, F11, Shift+F11, Backspace, Shift+Backspace, Ctrl+Shift+Delete, Delete, Enter, Shift+Enter, Ctrl+Z, Ctrl+Shift+Z, Ctrl+Y, and Ctrl+U. The code is as follows:1 [
2 { "keys": ["ctrl+shift+n"], "command": "new_file"
3 { "keys": ["ctrl+shift+w"], "command": "close_file"
4 { "keys": ["ctrl+o"], "command": "open_file"
5 { "keys": ["ctrl+shift+t"], "command": "revert_file"
6 { "keys": ["alt+o"], "command": "revert_file"
7 { "keys": ["ctrl+n"], "command": "revert_file"
8 { "keys": ["ctrl+s"], "command": "revert_file"
9 { "keys": ["ctrl+shift+s"], "command": "revert_file"
10 { "keys": ["ctrl+f4"], "command": "revert_file"
11 { "keys": ["ctrl+w"], "command": "revert_file"
12
13 { "keys": ["ctrl+k"], "command": "cancel_edit"
14 { "keys": ["f11"], "command": "cancel_edit"
15 { "keys": ["shift+f11"], "command": "cancel_edit"
16
17 { "keys": ["backspace"], "command": "cancel_edit"
18 { "keys": ["shift+backspace"], "command": "cancel_edit"
19 { "keys": ["ctrl+shift+backspace"], "command": "cancel_edit"
20 { "keys": ["delete"], "command": "cancel_edit"
21 { "keys": ["enter"], "command": "cancel_edit"
22 { "keys": ["shift+enter"], "command": "cancel_edit"
23
24 { "keys": ["ctrl+z"], "command": "cancel_edit"
25 { "keys": ["ctrl+shift+z"], "command": "cancel_edit"
26 { "keys": ["ctrl+y"], "command": "cancel_edit"
27 { "keys": ["ctrl+u"], "command": "cancel_edit"

The following example shows how to perform key bindings in Windows:

```
[  
  { "keys": ["ctrl+shift+n"], "command": "new_window" },  
  { "keys": ["ctrl+shift+w"], "command": "close_window" }  
]
```

Defining Key Bindings

Sublime Text editor includes an option to define a key map. The key bindings defined in the file **.sublime-keymap** includes all the key value combinations.



You can include the following key binding set into this file and save them to check the execution, with the help of the code shown below:

```
[{ "keys": ["super+alt+;"], "command": "run_macro_file",  
  "args": {"file": "Packages/User/semicolon.sublime-macro"} }]
```

Here **super** is the Winkey in Windows or Linux, and a command on OSX operating system. Note that this code will run the macro that is located in **Packages/User** and is named **semicolon.sublime-macro** on pressing the **Super+Alt+** keys.

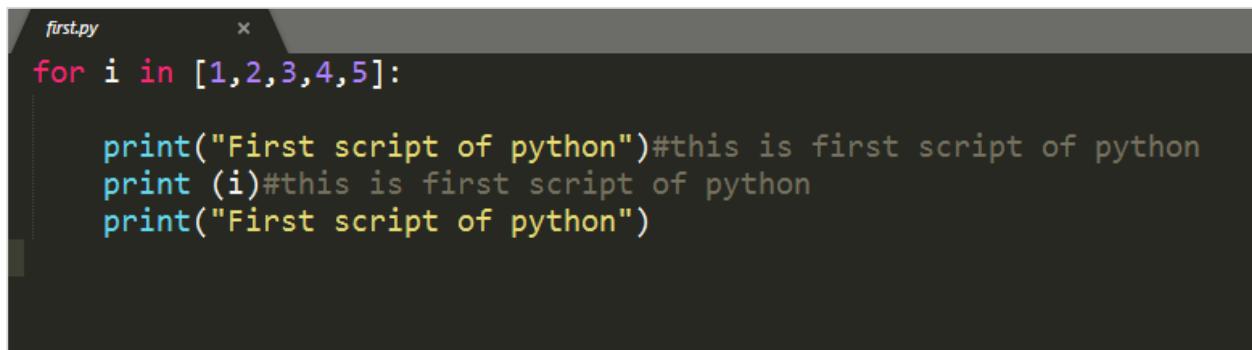
12. Sublime Text – Column Selection

In this chapter, you will learn various shortcuts keys to move and select the columns of a file in Sublime Text.

Moving the Pointer to the Beginning of Code

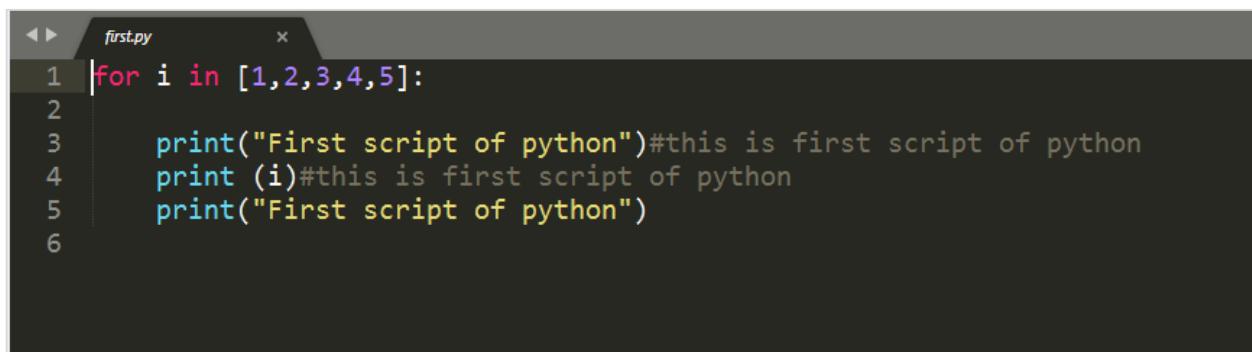
You can use the shortcut **Ctrl+Home** to move the pointer to the beginning of the code in Sublime Text editor.

Observe the sample code shown here.



```
first.py
for i in [1,2,3,4,5]:
    print("First script of python")#this is first script of python
    print (i)#this is first script of python
    print("First script of python")
```

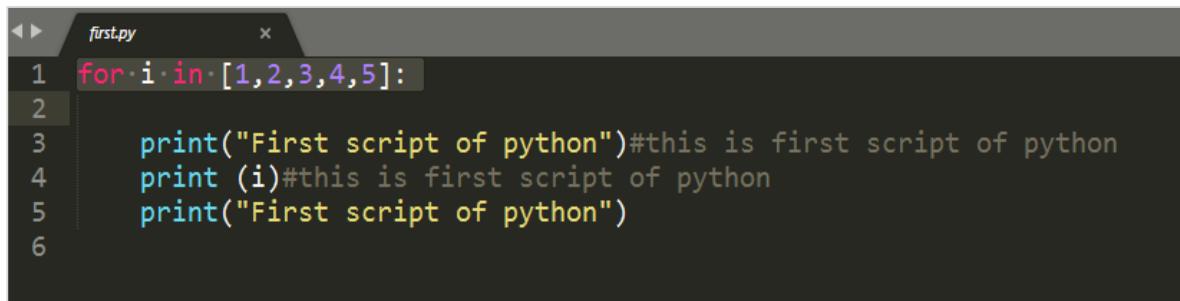
The screenshot given below shows the same code where the cursor is moved to its beginning.



```
first.py
1 |for i in [1,2,3,4,5]:
2
3     print("First script of python")#this is first script of python
4     print (i)#this is first script of python
5     print("First script of python")
6
```

Extending a Column

You can use the shortcut **Ctrl+L** to extend a specified column. This shortcut will select the column which includes the pointer. The following screenshot explains this in a detailed manner:



A screenshot of the Sublime Text editor showing a Python script named 'first.py'. The code is as follows:

```

1 for i in [1,2,3,4,5]:
2
3     print("First script of python")#this is first script of python
4     print (i)#this is first script of python
5     print("First script of python")
6

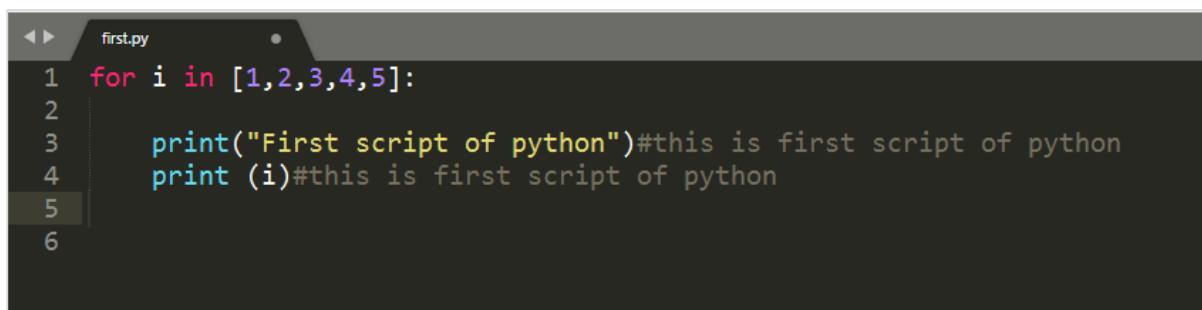
```

The cursor is positioned at the start of the fifth line, and the entire column containing the numbers 1 through 5 is selected, highlighted in light blue.

Trimming a column

You can use the shortcut **Ctrl+K** to trim the specified column of the file. For OSX, you will have to use the key combination **Cmd+K** for this purpose.

The example shown here displays the code where the line mentioned in the fifth column is trimmed.



A screenshot of the Sublime Text editor showing the same Python script 'first.py'. The code is identical to the previous screenshot, but the fifth column has been trimmed, resulting in the following output:

```

1 for i in [1,2,3,4,5]:
2
3     print("First script of python")#this is first script of python
4     print (i)#this is first script of python
5
6

```

Split a Block of Text

You can use the shortcut **Ctrl+Shift+K** to split a selected block of text into two selections.



A screenshot of the Sublime Text editor showing the Python script 'first.py'. The fifth column is selected, indicated by a vertical bar on the left margin and a horizontal selection bar across the code area.

```

1 for i in [1,2,3,4,5]:
2
3     print("First script of python")#this is first script of python
4     print (i)#this is first script of python
5
6

```

Note: The difference in key combinations from the previous key combination is that columns are visible. **Ctrl+Shift+K** removes the column number from the specified code, whereas **Ctrl+K** keeps the column number intact after trimming the entire column.

13. Sublime Text – Indentation

Indentation is the process of maintaining the code blocks of programming languages to maintain a systematic structure. It helps to convey the developers a better structure of program. This chapter explains you in detail about indentation in Sublime Text editor.

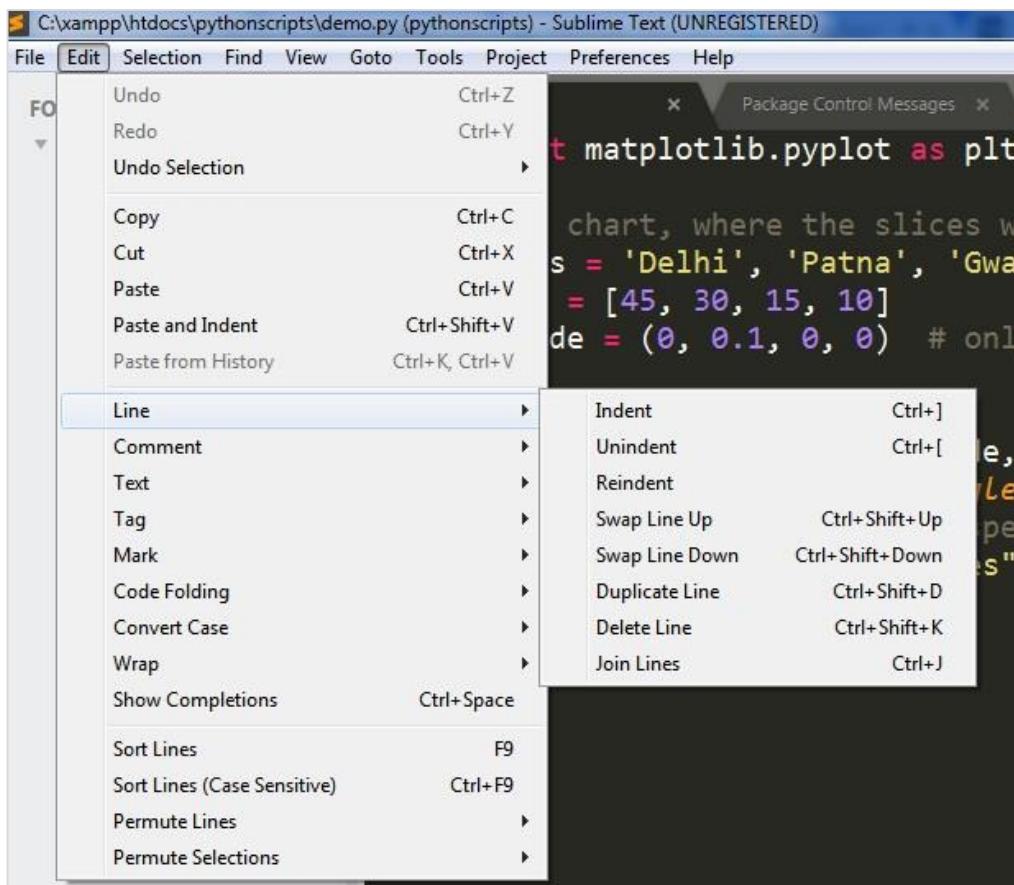
Importance of Indentation

Indentation helps to maintain the code structure such that it is easily understood by anyone who accesses it. Proper indentation imparts the following features to the code:

- Readability
- User-friendliness
- Adaptability
- Ease of maintenance

Options for Indentation in Sublime Text

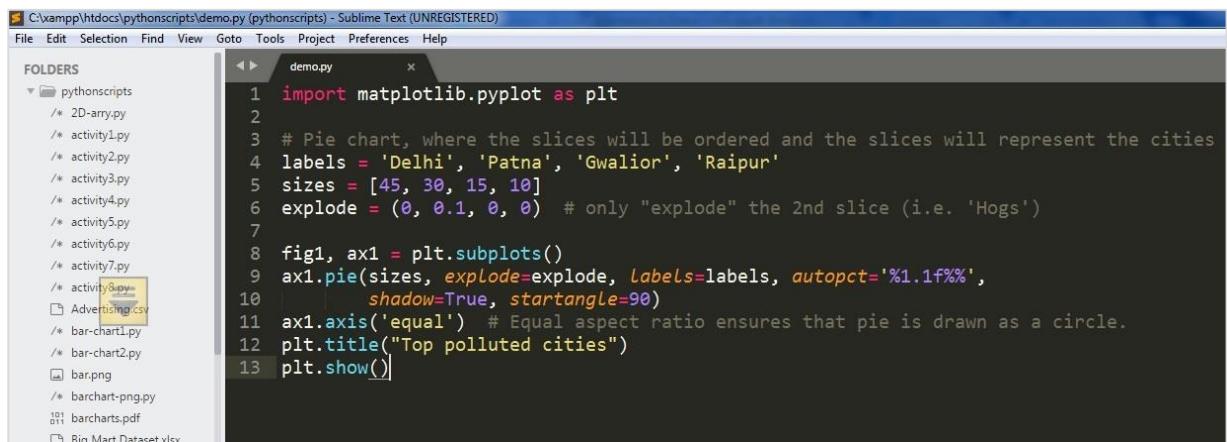
Sublime Text editor provides various options for code indentation as shown in the following screenshot:



The usage of these options is discussed in detail here:

Indent Option

Indent option helps to create indentation of the specified code. You have to select the code specified and select the **Indent** option. You can also use the shortcut **Ctrl+[** after selecting the required code that is to be indented. Refer to the following screenshot for a better understanding:



A screenshot of the Sublime Text editor. The title bar shows 'C:\xampp\htdocs\pythonscripts\demo.py (pythonscripts) - Sublime Text (UNREGISTERED)'. The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The left sidebar shows a file tree with 'FOLDERS' containing 'pythonscripts' and several Python files like '2D-arry.py', 'activity1.py', etc. A file named 'Advertising.csv' is highlighted with a yellow box. The main editor window displays the following Python code:

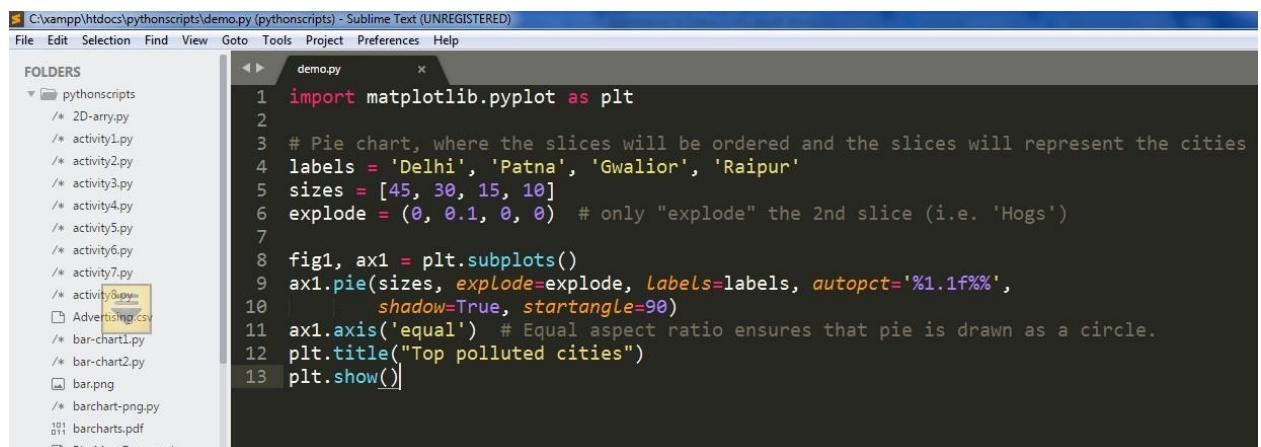
```

1 import matplotlib.pyplot as plt
2
3 # Pie chart, where the slices will be ordered and the slices will represent the cities
4 labels = 'Delhi', 'Patna', 'Gwalior', 'Raipur'
5 sizes = [45, 30, 15, 10]
6 explode = (0, 0.1, 0, 0) # only "explode" the 2nd slice (i.e. 'Hogs')
7
8 fig1, ax1 = plt.subplots()
9 ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
10         shadow=True, startangle=90)
11 ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.
12 plt.title("Top polluted cities")
13 plt.show()

```

Unindent Option

Unindent option works in the reverse of indent procedure. The shortcut for the unindent option is **Ctrl+]**. Refer to the following screenshot for a better understanding:



A screenshot of the Sublime Text editor, identical to the previous one but with a different selection. The code in the editor is now:

```

1 import matplotlib.pyplot as plt
2
3 # Pie chart, where the slices will be ordered and the slices will represent the cities
4 labels = 'Delhi', 'Patna', 'Gwalior', 'Raipur'
5 sizes = [45, 30, 15, 10]
6 explode = (0, 0.1, 0, 0) # only "explode" the 2nd slice (i.e. 'Hogs')
7
8 fig1, ax1 = plt.subplots()
9 ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
10         shadow=True, startangle=90)
11 ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.
12 plt.title("Top polluted cities")
13 plt.show()

```

Reindent Option

Reindent option is used to undo the unindent option and revert the code back to the same condition. Refer to the following screenshot for a better understanding:

```

C:\xampp\htdocs\pythonscripts\demo.py (pythonscripts) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
pythonscripts
  2D-arry.py
  activity1.py
  activity2.py
  activity3.py
  activity4.py
  activity5.py
  activity6.py
  activity7.py
  activity8.py
  Advertising.csv
  bar-chart1.py
  bar-chart2.py
  bar.png
  barchart-png.py
demo.py
1 import matplotlib.pyplot as plt
2
3 # Pie chart, where the slices will be ordered and the slices will represent the cities
4 labels = ['Delhi', 'Patna', 'Gwalior', 'Raipur']
5 sizes = [45, 30, 15, 10]
6 explode = (0, 0.1, 0, 0) # only "explode" the 2nd slice (i.e. 'Hogs')
7
8 fig1, ax1 = plt.subplots()
9 ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
10         shadow=True, startangle=90)
11 ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.
12 plt.title("Top polluted cities")
13 plt.show()

```

Benefits of Indentation

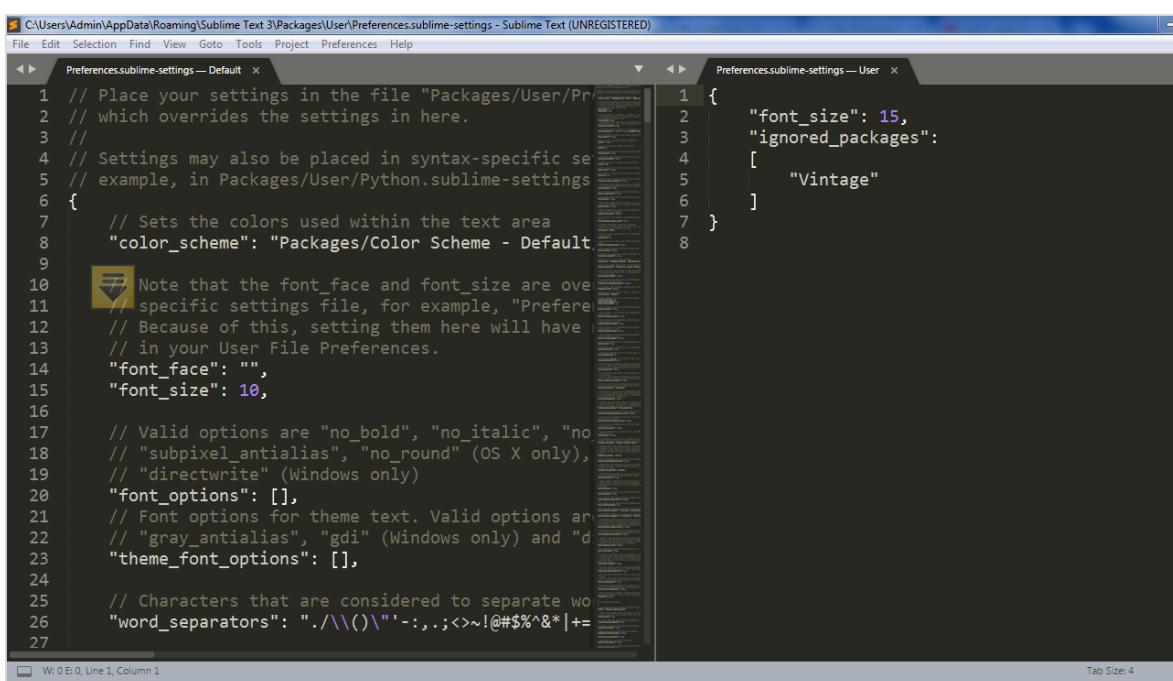
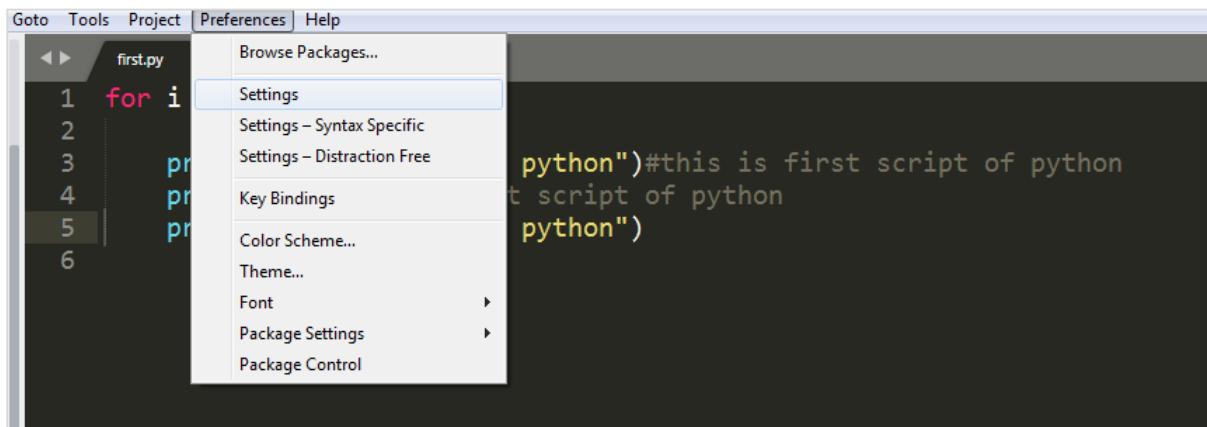
Proper indentation offers various benefits to the code and developers. They are discussed below:

- Code blocks will be visible without much effort.
- Developer can focus on lines of code that are necessary and easily ignore the irrelevant ones.
- A developer using the same kind of code structure multiple times need not focus on the **start of block** and **end of block** statements.
- Any misplaced lines of code can be easily identified.

14. Sublime Text – Base Settings

Base settings of the Sublime Text editor are the configuration needed for editor such as font size, display of line numbers or color scheme of the editor.

All the base settings are configured in JSON format. To view the list of settings, go to menu of **Preferences -> Settings**.



The settings include two types of configurations namely:

1. Default
 2. User

Default includes all the default configurations and user defines the customized settings for base settings.

For the scenario mentioned above, the customized settings are as mentioned below:

```
"font_size": 15,
    "ignored_packages":
    [
        "Vintage"
    ]
```

Categories of Settings

Various categories of the settings in Sublime Text are as follows:

Editor Settings

These include the basic settings for editing the files of the code base. Examples include **font_face**, **font_size** and **tab_size**. The settings are present in the default configuration.

User Interface Settings

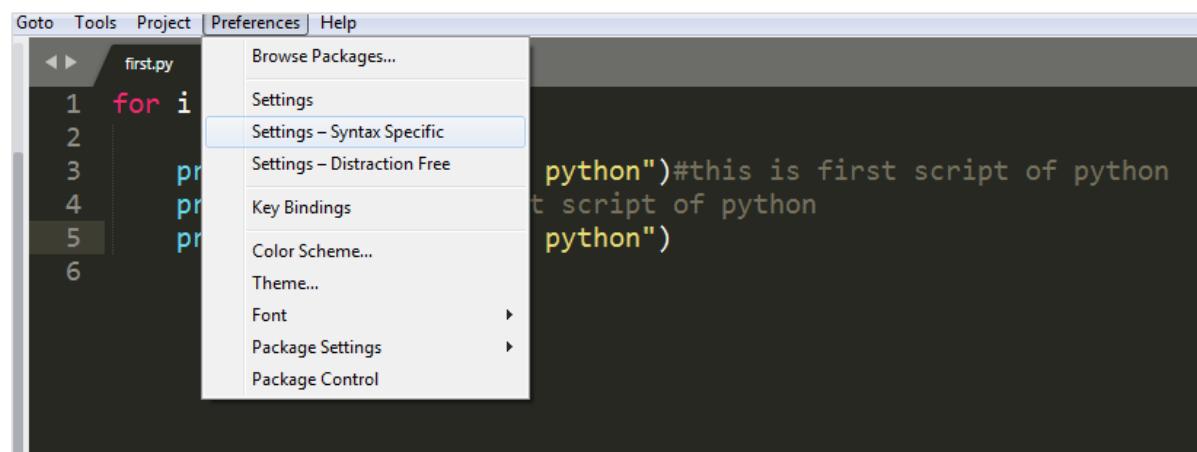
These include the general settings which specifically focus on background, theme and various color combinations. These settings are appended in the second section of the default configuration.

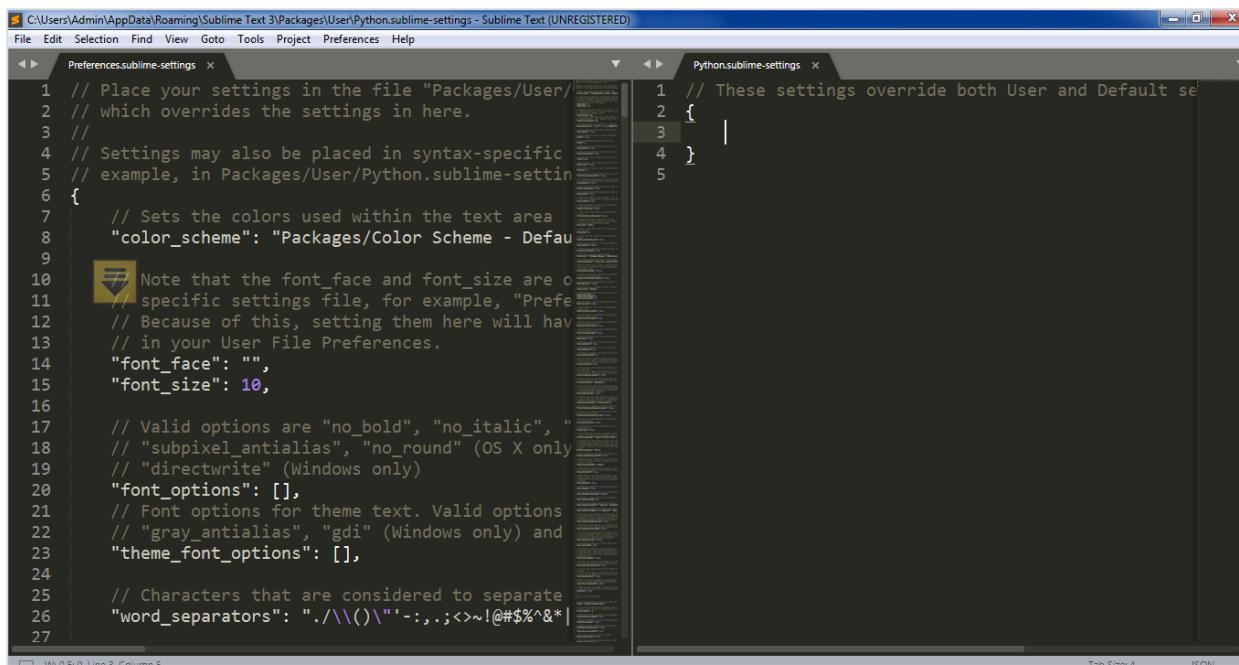
Application Behavior

These settings focus on behavior of the application included in Sublime Text editor across open windows. These settings are included in the third section of default configuration settings.

Syntax- Specific Settings

These settings are used for designing the syntax of Sublime Text editor. To get the list of syntax specific settings, the user needs to use option **Preferences-> Syntax-Specific**.





```
C:\Users\Admin\AppData\Roaming\Sublime Text 3\Packages\User\Python.sublime-settings - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Preferences.sublime-settings x
1 // Place your settings in the file "Packages/User/"
2 // which overrides the settings in here.
3 //
4 // Settings may also be placed in syntax-specific
5 // example, in Packages/User/Python.sublime-settings
6 {
7     // Sets the colors used within the text area
8     "color_scheme": "Packages/Color Scheme - Default"
9
10    // Note that the font_face and font_size are only
11    // specific settings file, for example, "Preferences.sublime-settings".
12    // Because of this, setting them here will have no effect
13    // in your User File Preferences.
14    "font_face": "",
15    "font_size": 10,
16
17    // Valid options are "no_bold", "no_italic", "no_strikethrough",
18    // "subpixel_antialias", "no_round" (OS X only)
19    // "directwrite" (Windows only)
20    "font_options": [],
21    // Font options for theme text. Valid options
22    // "gray_antialias", "gdi" (Windows only) and
23    "theme_font_options": [],
24
25    // Characters that are considered to separate words
26    "word_separators": "./\\()\"`-:,.;<>~!@#$%^&*|"
27
Python.sublime-settings x
1 // These settings override both User and Default settings
2 {
3     |
4 }
5
```

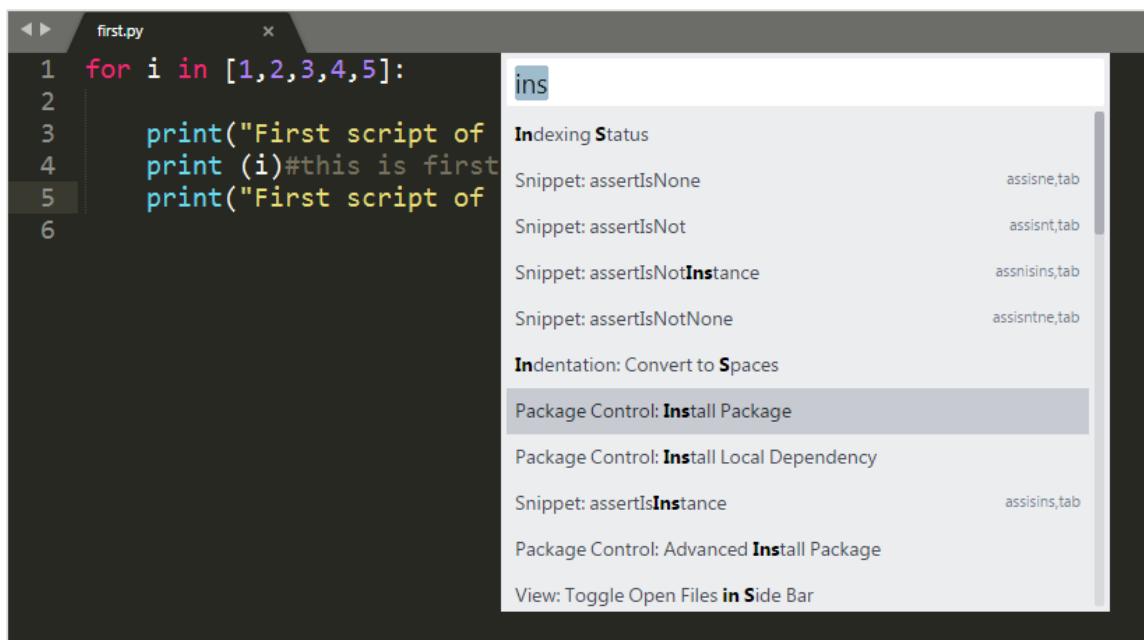
15. Sublime Text – Theme Management

Theme management in Sublime text editor implies enhancing the appearance of editor with colors and attractive backgrounds. Themes are JSON formatted values with **.sublime-theme** extension and manage the appearance of the user interface of the editor by changing the colors of elements of the IDE.

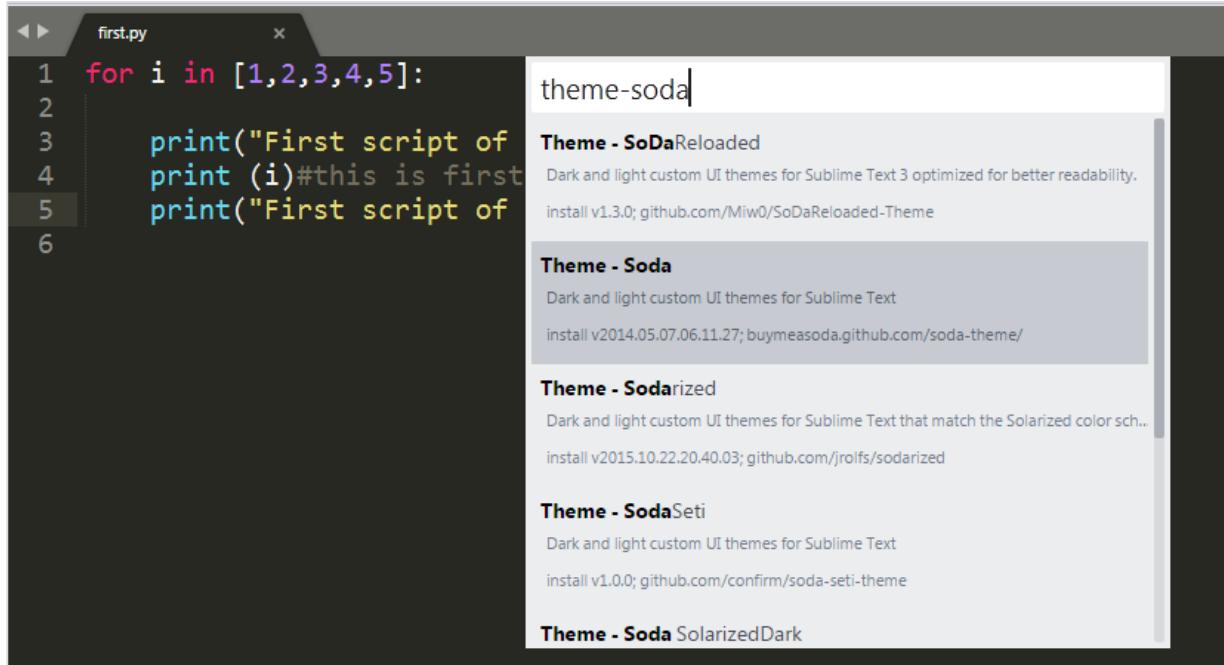
Installing and Applying Themes

The following steps will guide you about installing and applying themes in Sublime Text editor:

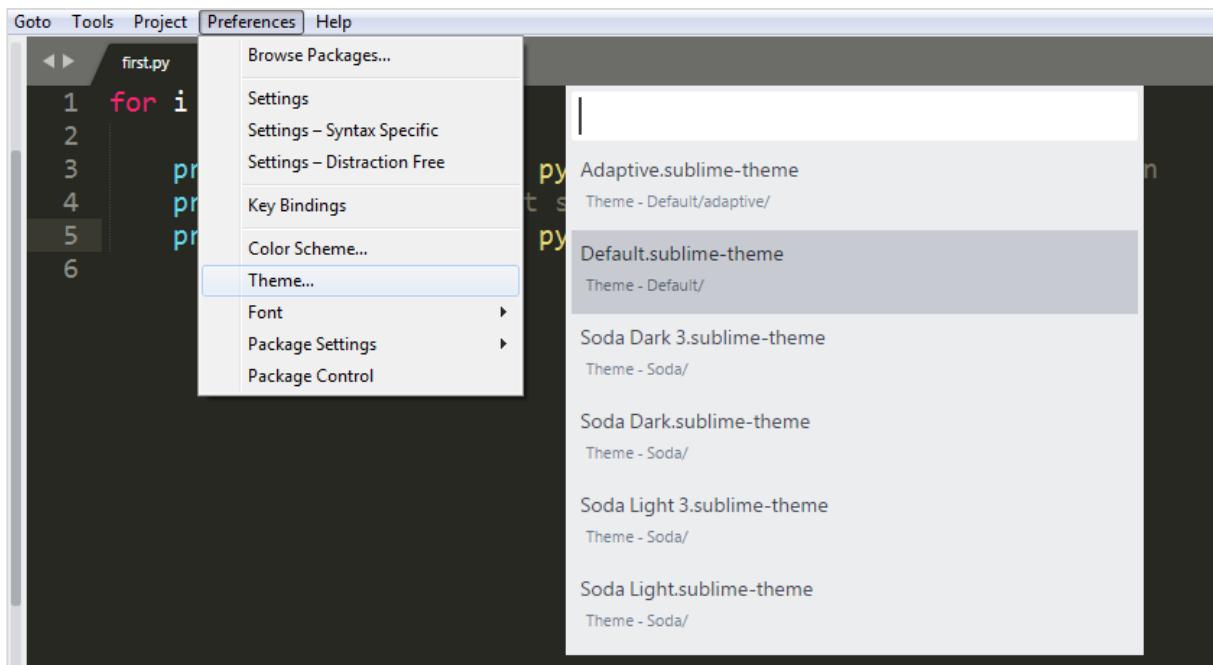
Step 1: Use the **Install** Package of the Sublime Text to install package for themes, as shown in the screenshot here:



Step 2: You can see a list of options with **Install Package** option and you can download them and install in Sublime Text editor.



Step 3: Once you select a theme, activate it.



Step 4: Now select your desired theme from the list of installed theme list. This automatically changes the appearance of the editor.

```

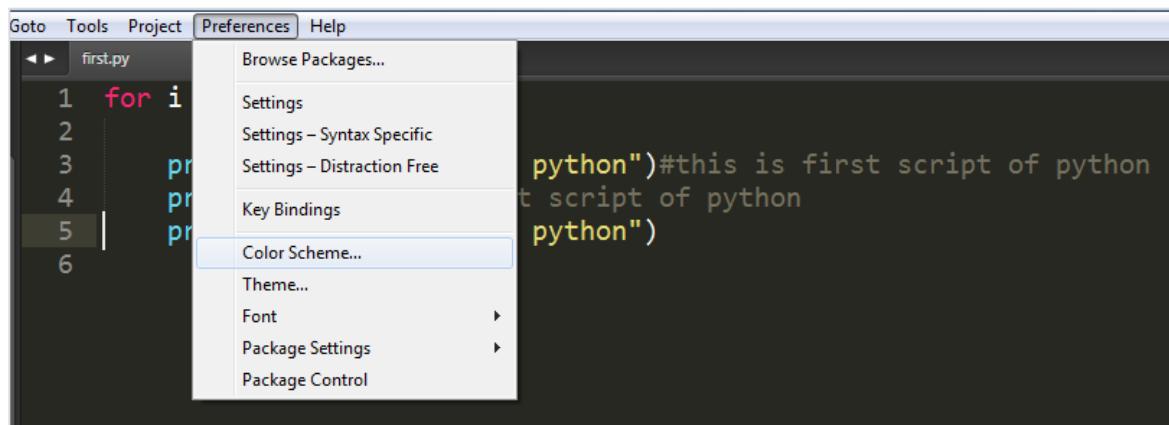
C:\xampp\htdocs\pythonscripts\first.py (pythonscripts) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
activity5.py
activity6.py
activity7.py
activity8.py
Advertising.csv
bar-chart.py
bar-chart2.py
bar.png
barchart-png.py
barcharts.pdf
Big Mart Dataset.xlsx
bubble-chart.py
central-tendencies.py
chart.py
contour-plot.py
create-png.py
data-munging.py
demo.py
demo1.py
demoloop.py
dispersion.py
first.html
first.py
functions.py
gradient-descent.py
hello-world.py
hypothesis.py
k-means.py
line-chart.py
log-plot.py
matrices.py
W:0 E:0 Line 5, Column 1

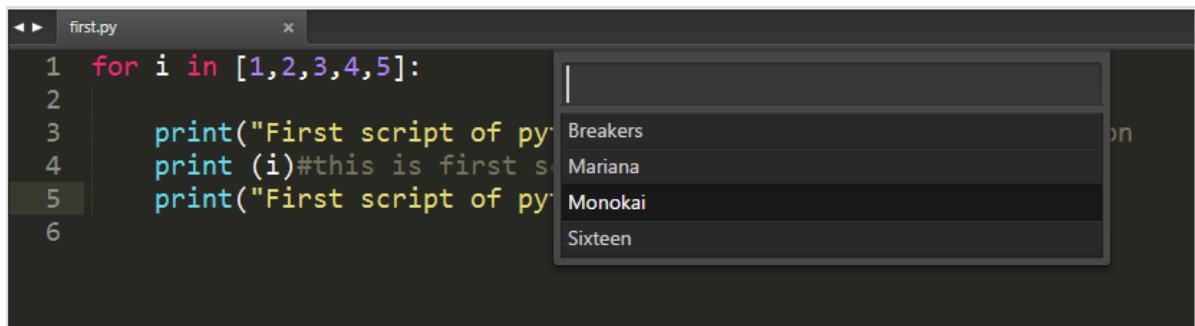
```

Color Schemes

Color schemes are XML formatted files located at the **Packages** folder with color schemes. They are an awesome way for the customization of Sublime text colors, unlike themes which are specific to UI elements.

You can choose color schemes using option **Preferences -> Color Scheme**. Refer to the following screenshot that shows the same.





Color schemes include various options such as **Breakers**, **Mariana**, **Monokai**, and **Sixteen**. The appearance of the editor depends on the color scheme that you choose. Note that these patterns will only focus on color syntax.

16. Sublime Text – Understanding Vintage Mode

Vintage mode in Sublime text editor is a package that includes a combination of editing features of vi. Vintage Mode lets you to use a list of vi commands in the Sublime editor.

Vintage mode is an open source project and you can download it from the link: <https://github.com/sublimehq/Vintage>

Understanding vi editor

Vi is a primitive and popularly used text editor. It was originally created for UNIX operating system and with basic functionality of scripts execution. It is a popularly used command line editor across various operating systems because it does not require mouse for operations.

Sublime includes **vim**, which is an advanced section of **vi** editor, and includes customization of macros, snippets and plugins.

Enabling Vintage Mode

You can enable Vintage mode through the following steps:

- Select the option **Preferences -> Settings**
- Edit the **json** file of the settings configuration with key name as **ignored_packages**. Vintage package is included in the **ignored_packages** section.

```
// List any packages to ignore here. When removing entries from this list,  
// a restart may be required if the package contains plugins.  
"ignored_packages": ["Vintage"]  
}
```

- Remove the attribute of Vintage from the configurations to enable Vintage property in Sublime text editor.

```
"ignored_packages": []
```

- Your vintage mode editor will appear as shown in the following screenshot:

```
61 +-- 3 lines: Is a socket 'connection oriented'?
62 static inline int connection_based(struct sock *sk)
63 +-- 3 lines: {
64     static int receiver_wake_function(wait_queue_t *wait, unsigned
65         void *key)
66     void *key)
67     10 lines: {
68         3 lines: Wait for a packet...
69     static int wait_for_packet(struct sock *sk, int *err, long *time)
70     45 lines: {
71         10 lines: {
72             3 lines: Wait for a packet...
73             static int wait_for_packet(struct sock *sk, int *err, long *time)
74             45 lines: {
75                 10 lines: {
76                     3 lines: Wait for a packet...
77                     static int wait_for_packet(struct sock *sk, int *err, long *time)
78                     45 lines: {
79                         10 lines: {
80                             3 lines: Wait for a packet...
81                             static int wait_for_packet(struct sock *sk, int *err, long *time)
82                             45 lines: {
83                                 10 lines: {
84                                     3 lines: Wait for a packet...
85                                     static int wait_for_packet(struct sock *sk, int *err, long *time)
86                                     45 lines: {
87                                         10 lines: {
88                                             3 lines: Wait for a packet...
89                                             static int wait_for_packet(struct sock *sk, int *err, long *time)
90                                             45 lines: {
91                                                 10 lines: {
92                                                     3 lines: Wait for a packet...
93                                                     static int wait_for_packet(struct sock *sk, int *err, long *time)
94                                                     45 lines: {
95                                                         10 lines: {
96                                                             3 lines: Wait for a packet...
97                                                             static int wait_for_packet(struct sock *sk, int *err, long *time)
98                                                             45 lines: {
99                                                                 10 lines: {
100                                     3 lines: Wait for a packet...
101                                     static int wait_for_packet(struct sock *sk, int *err, long *time)
102                                     45 lines: {
103                                         10 lines: {
104                                             3 lines: Wait for a packet...
105                                             static int wait_for_packet(struct sock *sk, int *err, long *time)
106                                             45 lines: {
107                                                 10 lines: {
108                                                     3 lines: Wait for a packet...
109                                                     static int wait_for_packet(struct sock *sk, int *err, long *time)
110                                                     45 lines: {
111                                                         10 lines: {
112                                                             3 lines: Wait for a packet...
113                                                             static int wait_for_packet(struct sock *sk, int *err, long *time)
114                                                             45 lines: {
115                                                                 10 lines: {
116                                                     3 lines: Wait for a packet...
117                                                     static int wait_for_packet(struct sock *sk, int *err, long *time)
118                                                     45 lines: {
119                                                         10 lines: {
120                                                             3 lines: Wait for a packet...
121                                                             static int wait_for_packet(struct sock *sk, int *err, long *time)
122                                                             45 lines: {
123                                                 10 lines: {
124                                                     3 lines: Wait for a packet...
125                                                     static int wait_for_packet(struct sock *sk, int *err, long *time)
126                                                     45 lines: {
127                                                         10 lines: {
128                                                             3 lines: Wait for a packet...
129                                                             static int wait_for_packet(struct sock *sk, int *err, long *time)
130                                                             45 lines: {
131                                                 10 lines: {
132                                                     3 lines: Wait for a packet...
133                                                     static int wait_for_packet(struct sock *sk, int *err, long *time)
134                                                     45 lines: {
135                                                         10 lines: {
136                                                             3 lines: Wait for a packet...
137                                                             static int wait_for_packet(struct sock *sk, int *err, long *time)
138
datagram.c                                              132,1      8%
1 +-- 6 lines: net/core/dst.c Protocol independent destination c
2
3 #include <linux/bits.h>
4 #include <linux/error.h>
5 #include <linux/init.h>
6 #include <linux/kernel.h>
7 #include <linux/workqueue.h>
8 #include <linux/mm.h>
9 #include <linux/module.h>
10 #include <linux/slab.h>
11 #include <linux/netdevice.h>
12 #include <linux/skbuff.h>
13 #include <linux/string.h>
14 #include <linux/types.h>
15 #include <net/net_namespace.h>
16 #include <linux/sched.h>
17
dst.c                                               1,1      100%
18
Top netpoll.c                                         645,30    39%
```

17. Sublime Text – Vintage Commands

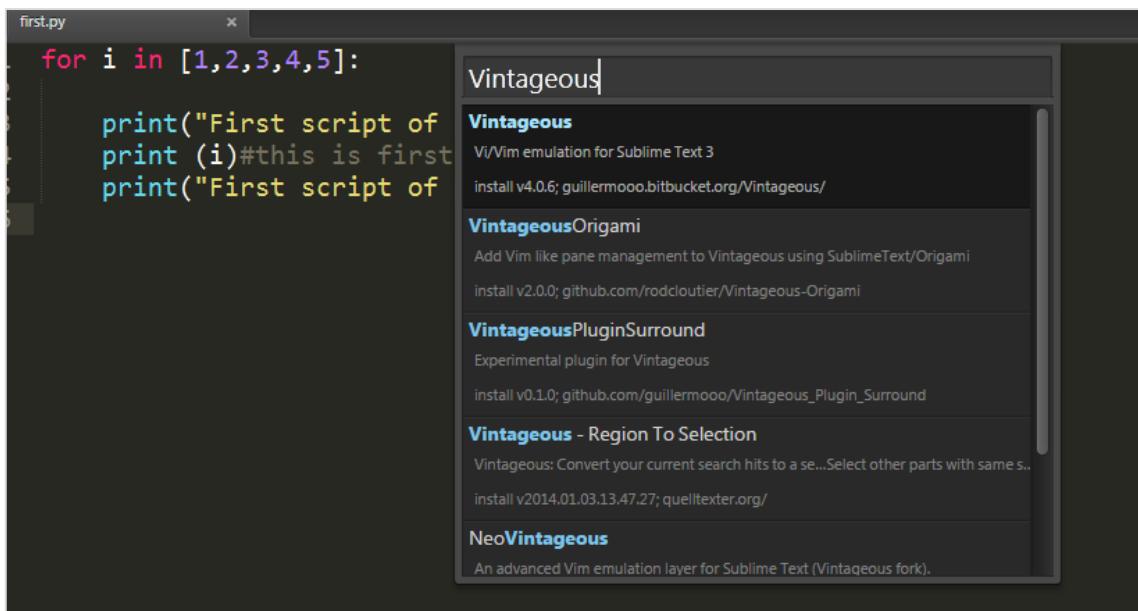
Vintage mode includes a variety of commands similar to the vi editor. This includes the basic settings of Sublime Text editor as shown in the code given below:

```
{  
    "color_scheme": "Packages/Color Scheme - Default/Monokai.tmTheme",  
    "font_size": 15,  
    "ignored_packages":  
    [  
  
    ],  
    "vintage_start_in_command_mode": true,  
    "theme": "Soda Dark.sublime-theme"  
}
```

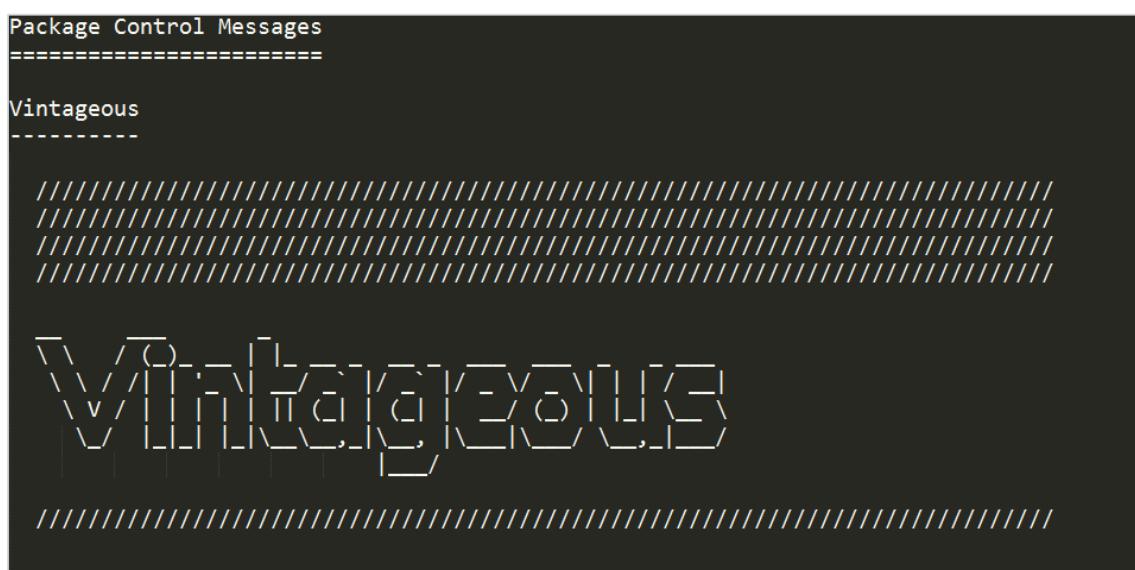
Mode	Description	Key
Command Mode	Waits for the user to enter a command	<i>Esc</i>
Insert Mode	Text can be inserted in different positions	<i>i/I/a/A</i>
Visual Mode	Select/highlight the text using the Movement Commands	<i>V</i>
Visual Line Mode	Select/highlight lines of text using the arrow keys	<i>Shift + V</i>

Vintageous Mode

Vintageous is the package used for the descriptive analysis of vi editor. It can be downloaded using package control as mentioned below:



Once you successfully install the vintageous package, your screen will look like as shown in the following screenshot:



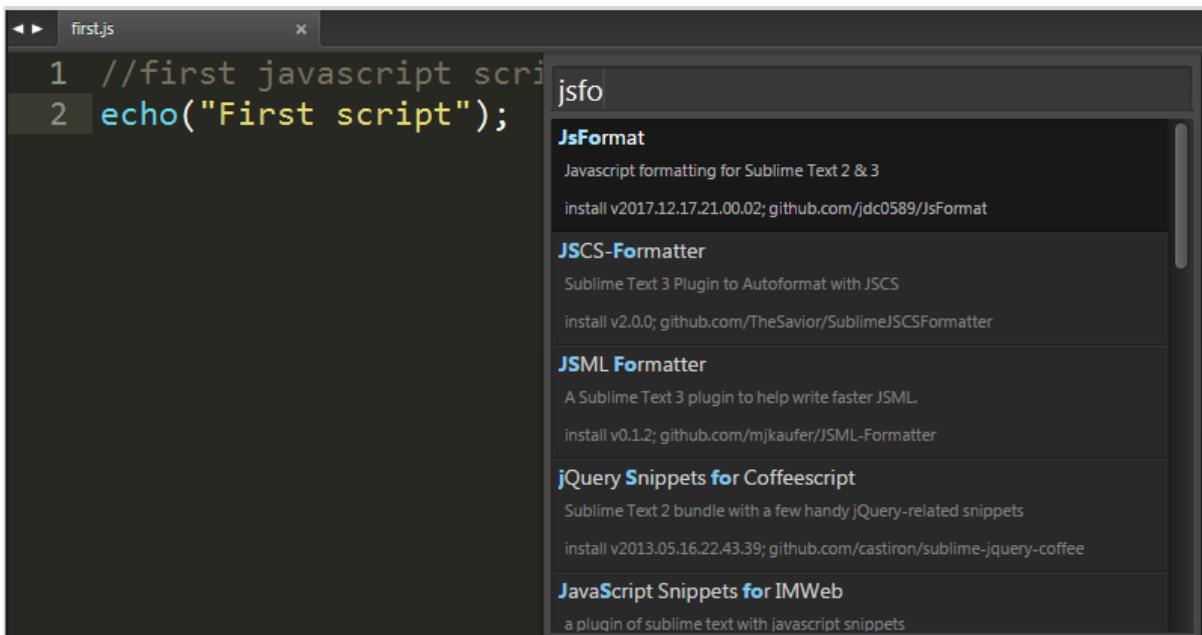
18. Sublime Text – Testing JavaScript

Sublime Editor includes testing modules for various programming languages. This helps in unit testing of files and scripts and helps developers to analyze bugs, errors and issues, if any.

Sublime Text editor includes 3 modules which are necessary for testing and deployment of JavaScript. This chapter discusses them in detail.

JsFormat

JsFormat is a JavaScript plugin used for formatting the script lines which eases the unit testing process. Behind the scenes, it uses JS beautifier (<http://jsbeautifier.org/>) to format the full JS or portions of JS files. JsFormat can be installed using the **Install Package** option of Sublime editor.



Features

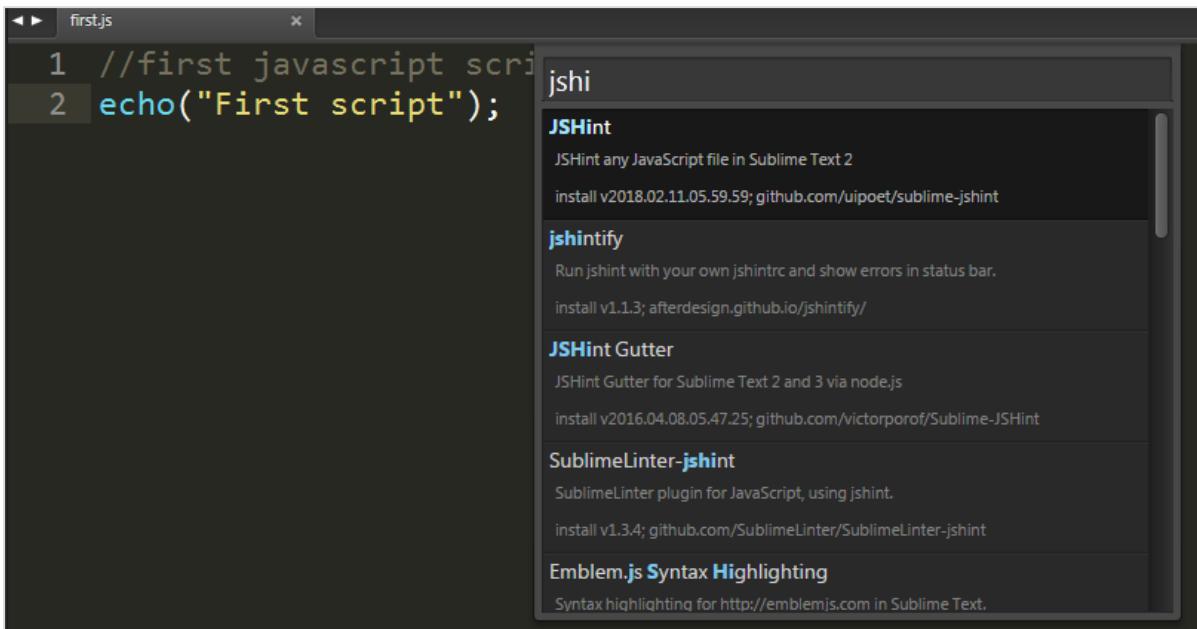
JsFormat offers the following features to the script:

- Eases JS and JSON file formatting.
- Offers full text formatting and selected formatting.
- Provides customizable settings for formatting options.

JSHint

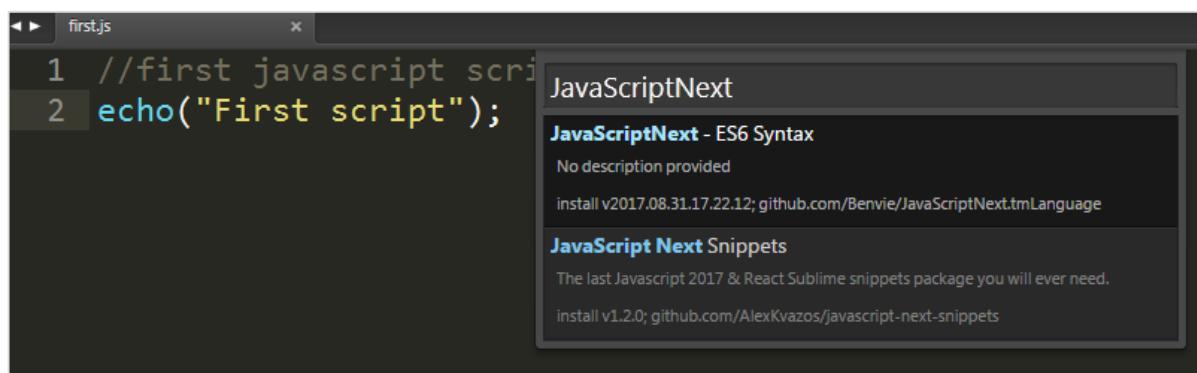
JSHint is a community driven tool used for analyzing the mistakes through hints. This helps to detect errors and potential problems. JSHint is an open source package, simple and easy to understand. You can install JSHint in Sublime Text editor through **Install Package** mode.

To implement JSHint plugin in Sublime Text Editor, you can use the shortcut **Alt+J** on Windows and **Cmd+j** on OSX systems.

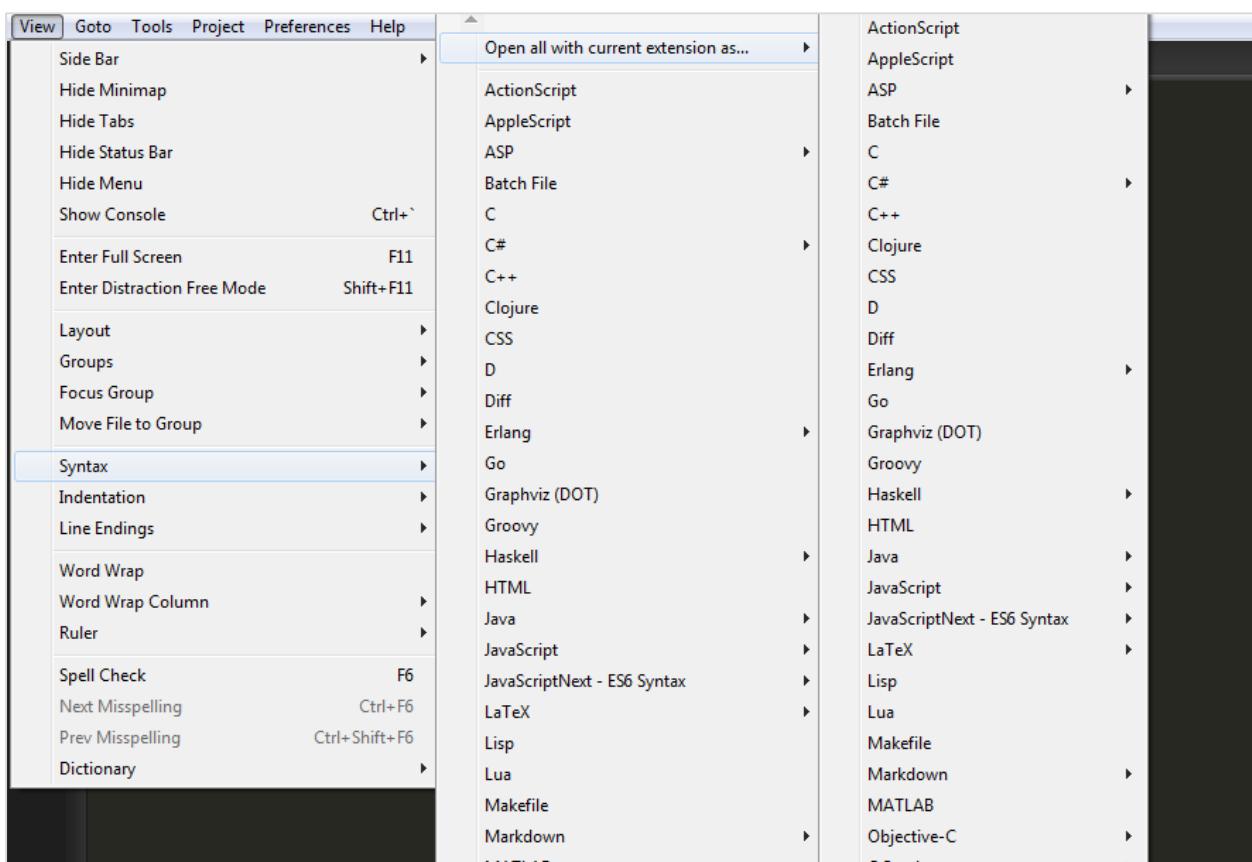


JavaScriptNext

This script is used to highlight the syntax of JS file and its associated meaning. It includes a feature of syntax highlight with various modules, arrow functions, classes and generators.



You can perform syntax check on a file using the syntax list as shown in the screenshot below:



19. Sublime Text – Testing Python Code

Python offers **unittest**, an official unit testing framework for unit testing the scripts designed before deployment. It is also called as **PyUnit**. Python unit tests used in Sublime Text editor are called as **sublime-unittests** and they are available in the following link:

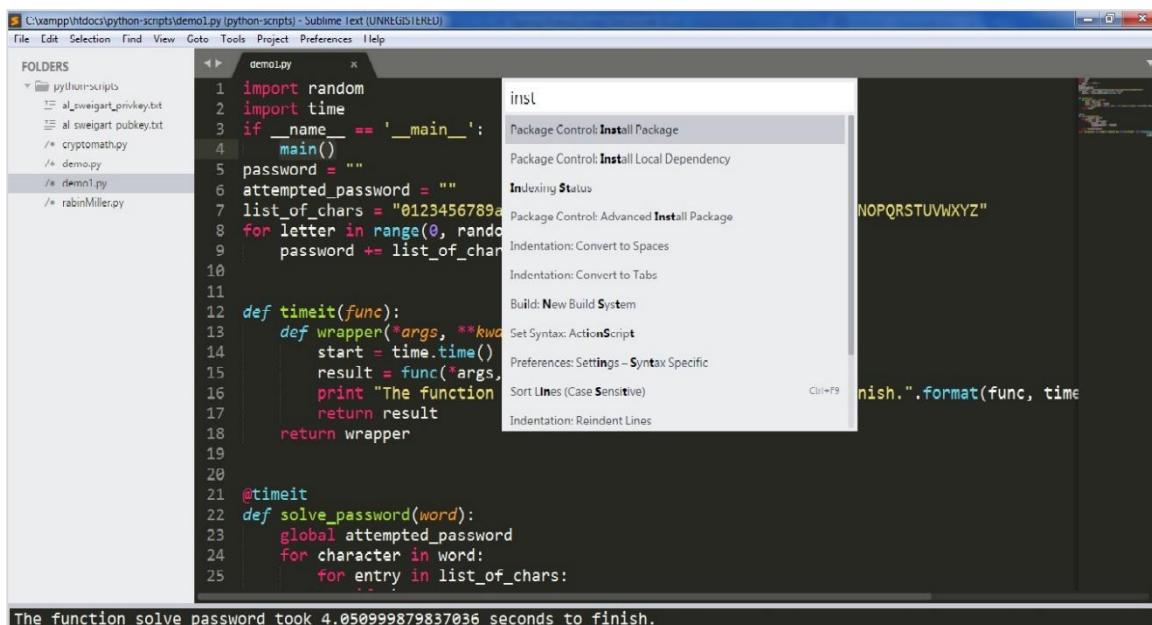
<https://github.com/martinsam/sublime-unittest>

These unit tests contain a number of useful snippets to ease the writing of unit test cases.

Installation of UnitTest

The package control of Sublime manages the installation of **unittests** and the following steps describe it in detail:

Step 1: Use the command palette of Sublime Editor **Ctrl+Shift+P** for installation of the package, as shown in the screenshots given below:



The screenshot shows a Sublime Text window with the title bar "C:\xampp\htdocs\python-scripts\demo1.py (python-scripts) - Sublime Text (UNKtG51tRtU)". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help. A sidebar on the left lists "FOLDERS" and files like "al_sweigart_privkey.txt", "al_sweigart_pubkey.txt", "cryptomath.py", "demo.py", "demo1.py", and "rabinMiller.py". The main editor area contains Python code for a password cracking script. A command palette is open at the bottom, with "instl" typed into the search field. The results list "Package Control: Install Package" as the top item, followed by "Package Control: Install Local Dependency", "Indexing Status", "Package Control: Advanced Install Package", "Indentation: Convert to Spaces", "Indentation: Convert to Tabs", "Build: New Build System", "Set Syntax: ActionScript", "Preferences: Settings – Syntax Specific", "Sort Lines (Case Sensitive)", and "Indentation: Reindent Lines". At the bottom of the editor, a status message reads "The function solve_password took 4.050999879837036 seconds to finish."

```

C:\xampp\htdocs\python-scripts\demo1.py (python-scripts) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
python-scripts
al_sweigart_privkey.txt
al_sweigart_pubkey.txt
cryptomath.py
demo.py
demo1.py
rabinMiller.py

demo1.py
1 import random
2 import time
3 if __name__ == '__main__':
4     main()
5 password = ""
6 attempted_password = ""
7 list_of_chars = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
8 for letter in range(0, random.randint(100000, 250000)):
9     password += list_of_chars[random.randint(0, 61)]
10
11 def timeit(func):
12     def wrapper(*args, **kw):
13         start = time.time()
14         result = func(*args, **kw)
15         print "The function"
16         print result
17         return result
18     return wrapper
19
20
21 @timeit
22 def solve_password(word):
23     global attempted_password
24     for character in word:
25         for entry in list_of_chars:
OPQRSTUVWXYZ"
ish.".format(func, time

```

The function solve_password took 4.050999879837036 seconds to finish.

```

C:\xampp\htdocs\python-scripts\demo1.py (python-scripts) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
python-scripts
al_sweigart_privkey.txt
al_sweigart_pubkey.txt
cryptomath.py
demo.py
demo1.py
rabinMiller.py

demo1.py
1 import random
2 import time
3
4 password = ""
5 attempted_password = ""
6 list_of_chars = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
7 for letter in range(0, random.randint(100000, 250000)):
8     password += list_of_chars[random.randint(0, 61)]
9
10
11 def timeit(func):
12     def wrapper(*a, **kw):
13         start = time.time()
14         result = func(*a, **kw)
15         print "The function"
16         print result
17         return result
18     return wrapper
19
20
21 @timeit
22 def solve_password(word):
23     global attempted_password
24     for character in word:
25         for entry in list_of_chars:
if character == entry:

```

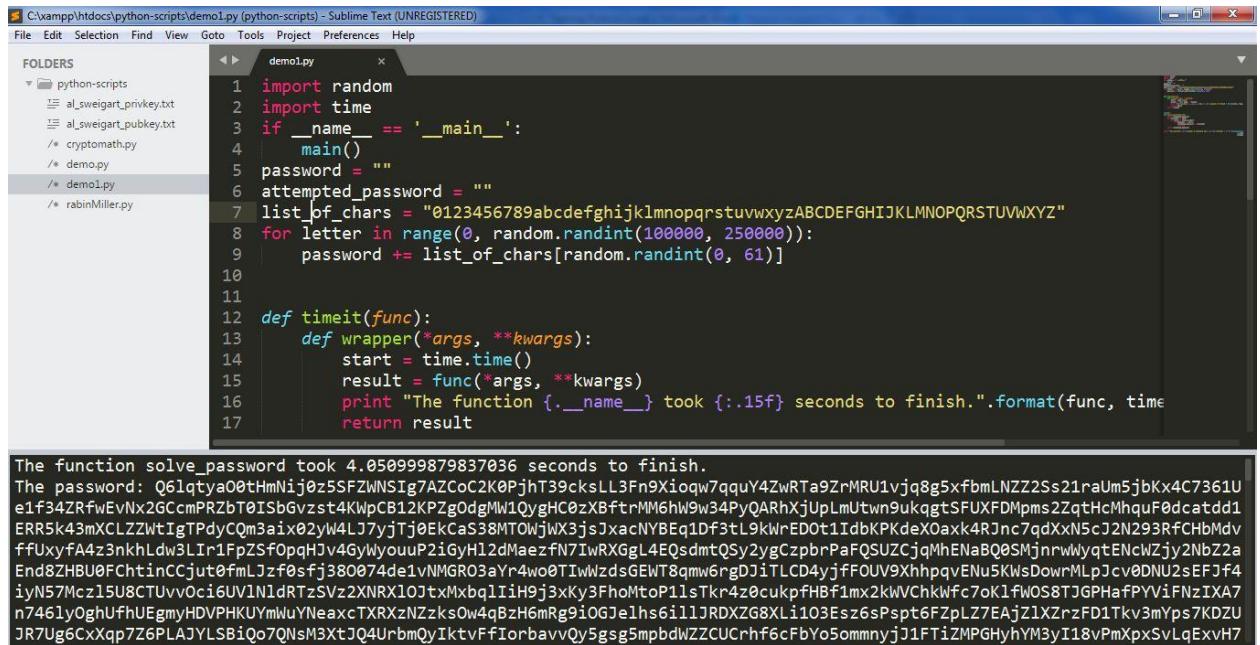
The function solve_password took 4.050999879837036 seconds to finish.

Step 2: The installed packages use the following two main snippets:

- **testclass** which is used to create a new test class
- **testfunc** which is used to create a new test function for us to fill out

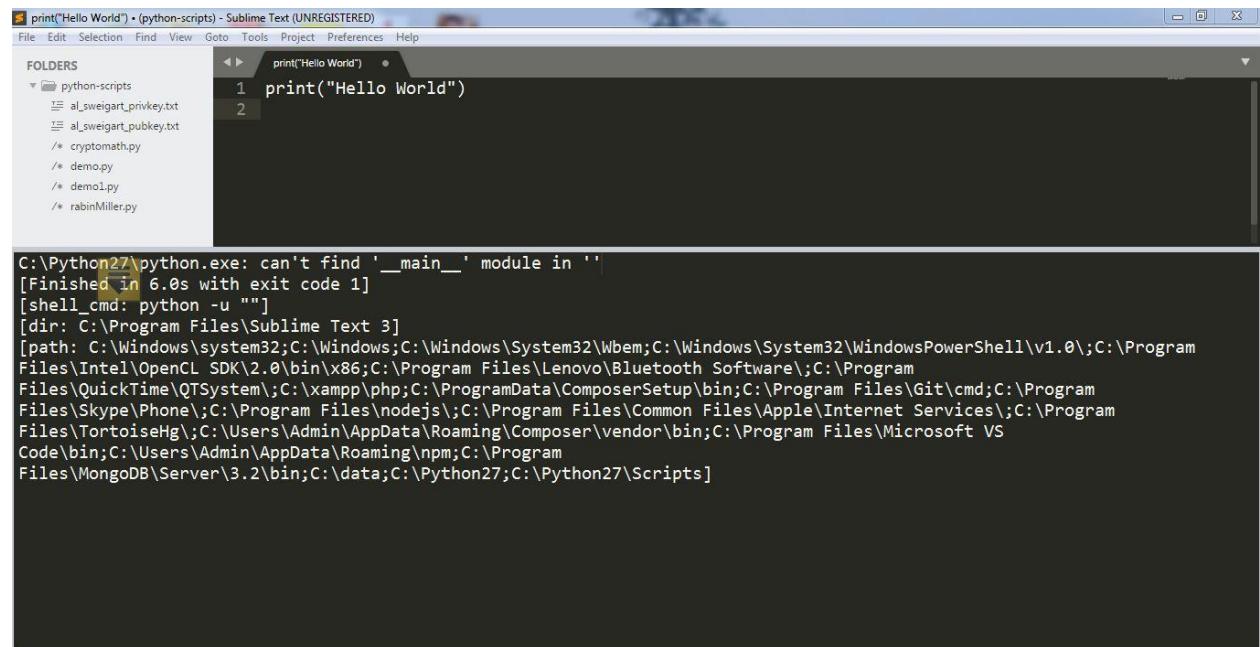
Step 3: The console computes test results as shown in the screenshot below. Note that the result depends on the success or failure of the test.

Success Result



```
C:\xampp\htdocs\python-scripts\demo1.py (python-scripts) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
python-scripts
al_sweigart_privkey.txt
al_sweigart_pubkey.txt
cryptomath.py
demo.py
demo1.py
rabinMiller.py
demo1.py
1 import random
2 import time
3 if __name__ == '__main__':
4     main()
5     password = ""
6     attempted_password = ""
7     list_of_chars = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
8     for letter in range(0, random.randint(100000, 250000)):
9         password += list_of_chars[random.randint(0, 61)]
10
11
12 def timeit(func):
13     def wrapper(*args, **kwargs):
14         start = time.time()
15         result = func(*args, **kwargs)
16         print "The function {:.__name__} took {:.15f} seconds to finish.".format(func, time
17
The function solve_password took 4.050999879837036 seconds to finish.
The password: Q6lqtya0tHmNij0z5SFZWSNSig7AZCoC2K0PjhT39cksLL3Fn9Xioqw7qquY4ZwRTa9ZrMRU1vjq8g5fbmLNZZ2Ss21raUm5jbKx4C7361U
e1f34ZRfwEvNx2GcmPRZbT0ISbGvzst4KwpCB12kPZg0dgW1QyghC0zXBfrMMGhW9w34Py0ARhXjUpLmUtn9ukqgtSFUXFDMpms2ZqthCmhqf0dcatdd1
ERR5k43mXCLZZWtIgTPdyCQm3aix02yW4LJ7yjtj0EkCaS38MT0WjWx3jsJxacNYBEq1Df3tL9kWnED0t1dbKPkdxOaxk4RJnc7qdXxN5cJ2N293RfcHbMdV
ffUxyfA4z3nkhLdw3L1r1FpZSf0pqhJv4gylyouuP2i6yH12dMaezfn7IwRXGgl4EQsdtmtQSy2ygCzpbPafQSUZCj0MhENaB00SMjnrwlyqtENclZjy2NbZ2a
End8ZHBu0FChinCCjut0fmlJzf0sfj380074de1vNMGR03aYr4wo0TIwWzsGEWT8qmw6rgDji1LCD4yjfFOUV9XhhpqvEnu5KWsDowrMLpJcv0DNU2sEFJf4
iyn57Mcz15U8CTUvvOci6UV1N1dRTzSVz2XNRX10JtxMxbqlIiH9j3xKy3fhoMtoP11sTkr4z0cukpfHbf1mx2kWVChkWfc7ok1fWOS8TJGPhafPYViFnzIXA7
n7461yOghUfhUEgmyHDVPHKUYmWuYNeaxcTXRXzNZZks0w4qBzH6mRg9iOGJe1hs6illJRDZG8XLi103Esz6sPspt6FZpLZ7EAjZ1XrzFD1Tkv3mYps7KDZU
J7Ug6CxXqp7Z6PLAJYLSBiQo70NsM3XtJQ4UrbdmQyIktrvffIorbavvQy5gsg5mpbdwZZCUCrhf6cFby5ommnyjJ1FTizMPGHyhYM3y118vPmXpxSvLqExvH7
```

Failure Result



```
C:\Python27\python.exe: can't find '__main__' module in ''
[Finished in 6.0s with exit code 1]
[shell_cmd: python -u ""]
[dir: C:\Program Files\Sublime Text 3]
[path: C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program
Files\Intel\OpenCL SDK\2.0\bin\x86;C:\Program Files\Lenovo\Bluetooth Software\;C:\Program
Files\QuickTime\QTSystem\;C:\xampp\php;C:\ProgramData\ComposerSetup\bin;C:\Program Files\Git\cmd;C:\Program
Files\Skype\Phone\;C:\Program Files\nodejs\;C:\Program Files\Common Files\Apple\Internet Services\;C:\Program
Files\TortoiseHg\;C:\Users\Admin\AppData\Roaming\Composer\vendor\bin;C:\Program Files\Microsoft VS
Code\bin;C:\Users\Admin\AppData\Roaming\npm;C:\Program
Files\MongoDB\Server\3.2\bin;C:\data;C:\Python27;C:\Python27\Scripts]
```

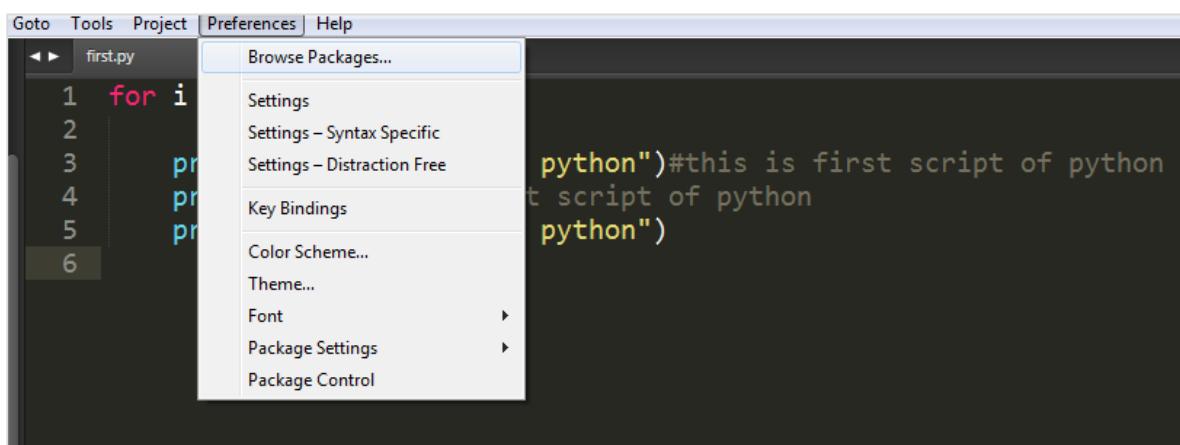
Note: Unit test cases work as a compiler for computing scripts in python.

20. Sublime Text – Spell Check

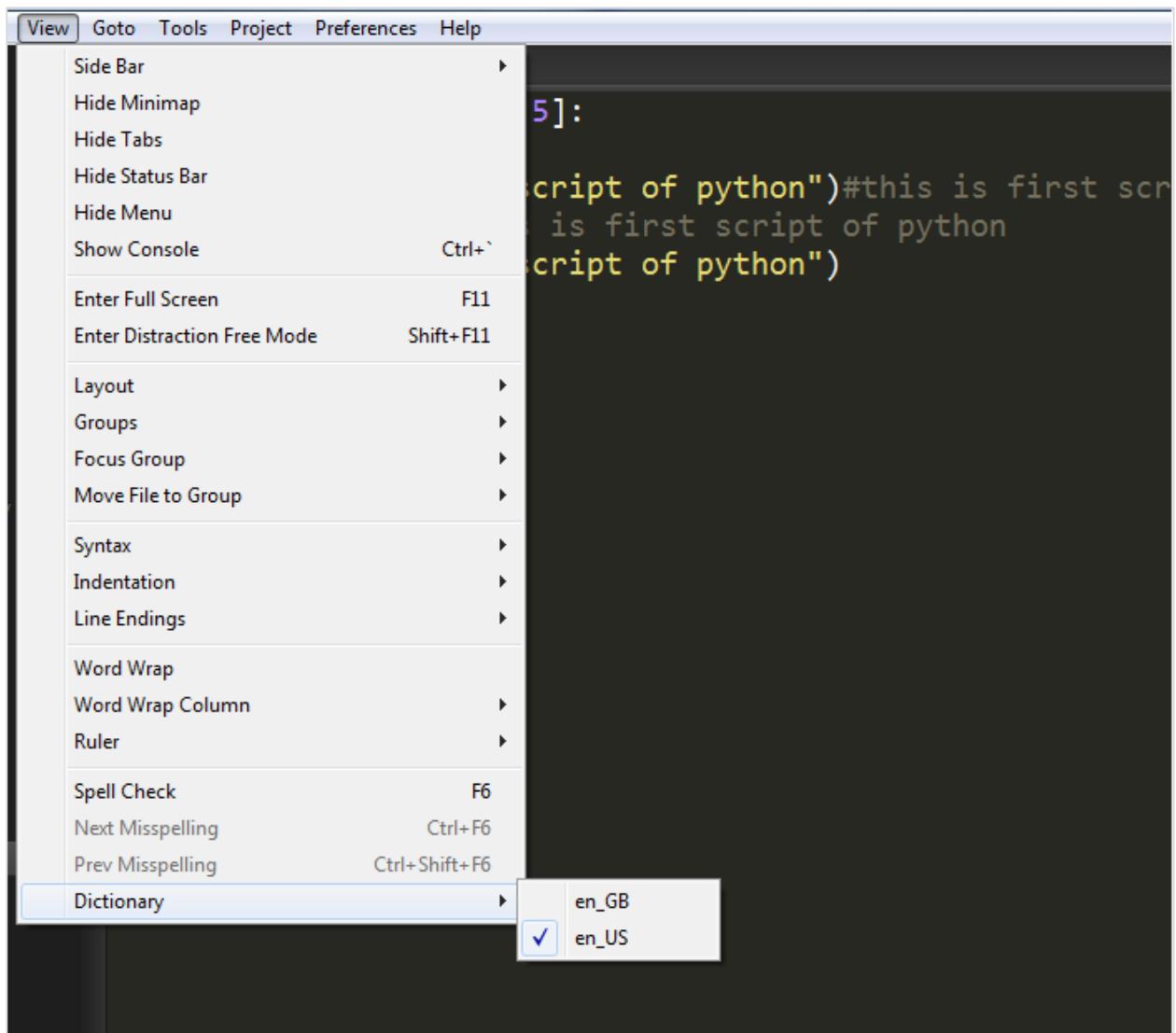
Sublime Text Editor uses **Hunspell** for spell checking process. **Hunspell** is the spell checker of LibreOffice, Mozilla Thunderbird, Google chrome and many proprietary packages. Sublime Text editor includes dictionary support for proper spell check of words.

Dictionaries

Sublime Text includes UTF-8 encoded dictionaries. To implement a dictionary with Sublime Text editor, it should be first converted into UTF-8. If the user has UTF-8 encoded dictionary, it can be installed using the **Preferences** option in Sublime Text editor.



You can select the dictionary from **View-> Dictionary** menu as shown in the given screenshot:



Settings for Dictionary

There are two settings defined for the spell check of Sublime Text Editor:

- Spell_check
- Dictionary

```
// Set to true to turn spell checking on by default
"spell_check": false,

// Word list to use for spell checking
"dictionary": "Packages/Language - English/en_US.dic"
```

These configurations are included in the **settings** file. Added and ignored words are stored in the user settings under the **added_words** and **ignored_words** keys, respectively.

21. Sublime Text – Packages

Software packages are installed in Sublime Text for additional functionalities. You can see the list of software packages through the shortcut key **Ctrl+Shift+P** on Windows and **Cmd+Shift+P** on Mac. Observe the following screenshots for a better understanding:

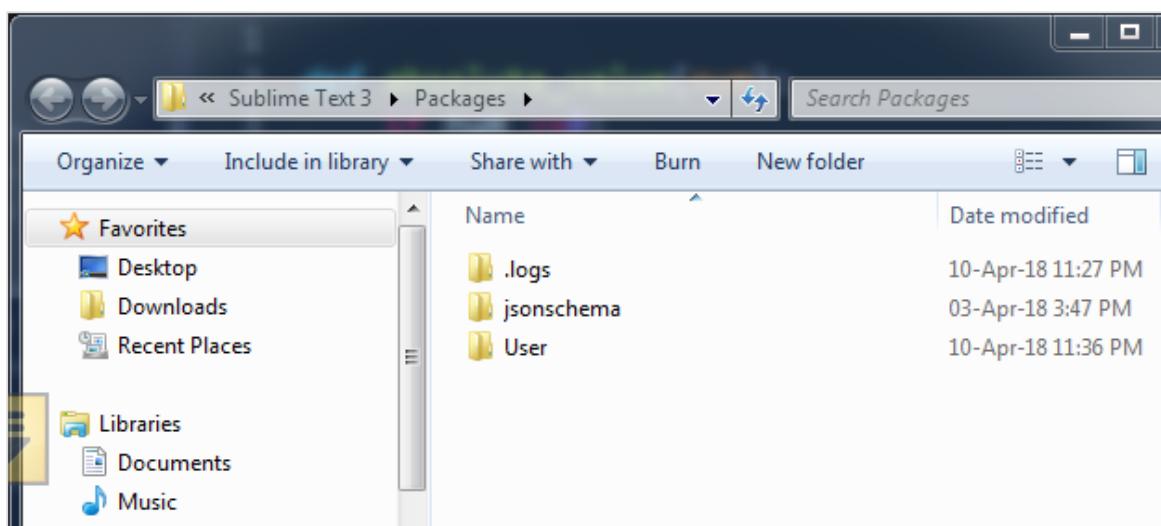
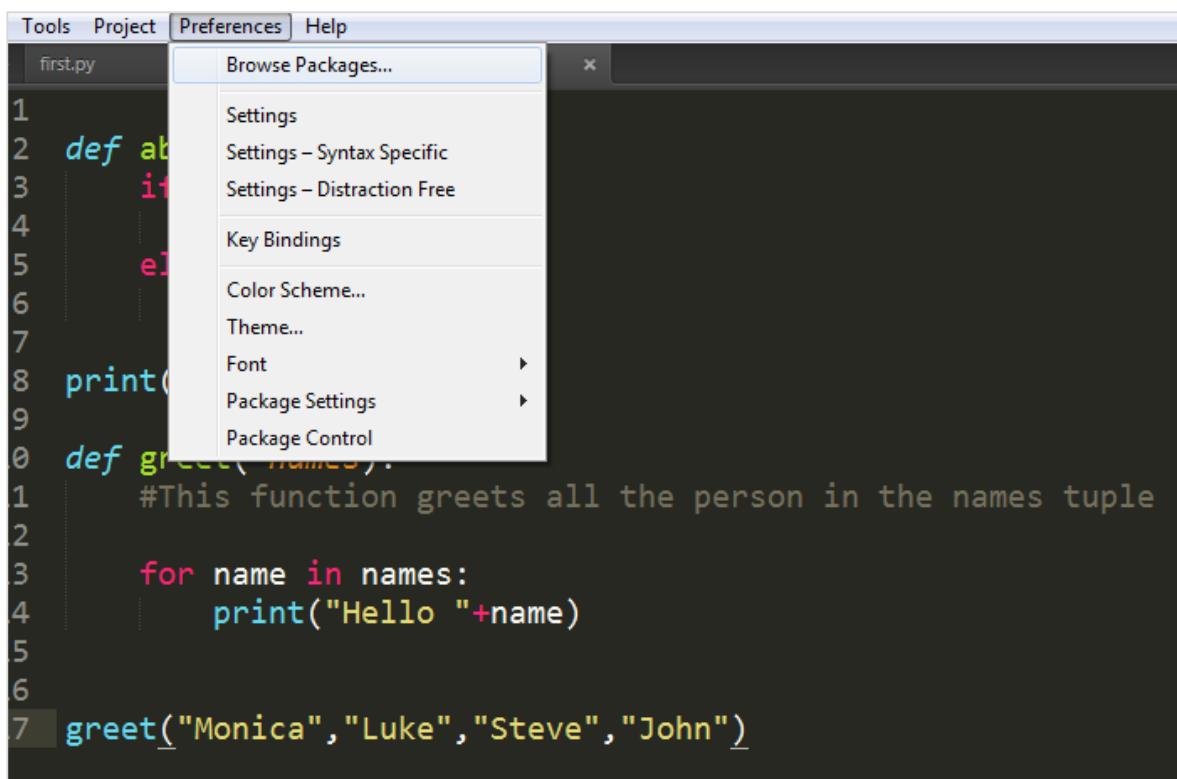
The screenshot shows the Sublime Text interface with two tabs open: 'first.py' and 'functions.py'. A command palette is open on the right side, displaying a list of packages and commands starting with 'inst'. The list includes:

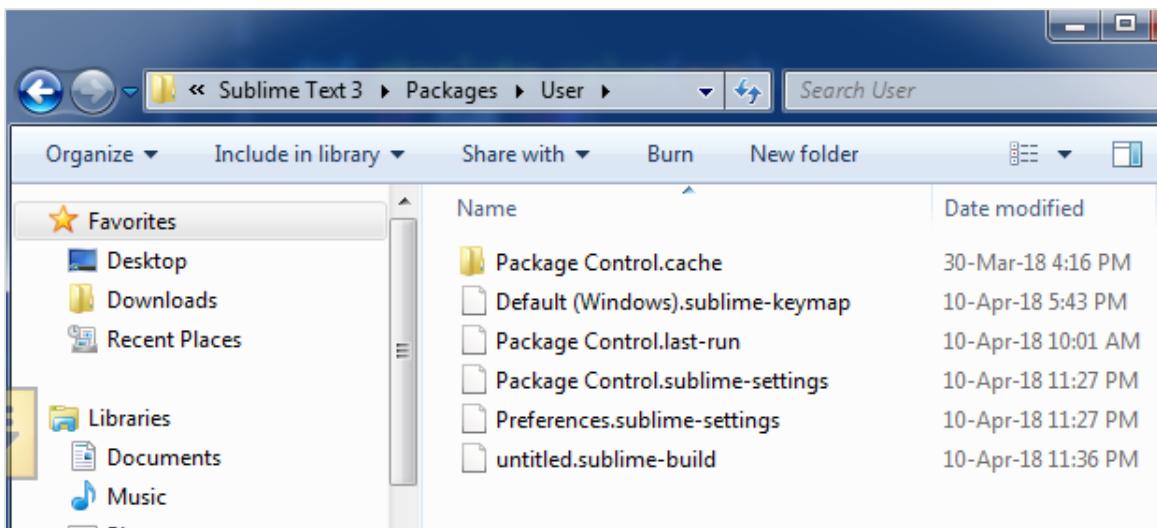
- Package Control: Install Package
- Package Control: Install Local Dependency
- Indexing Status
- Snippet: assertIsInstance
- Package Control: Advanced Install Package
- Snippet: assert IsNotInstance
- Indentation: Convert to Spaces
- Snippet: assert IsNot
- Snippet: assert IsNotNone
- Indentation: Convert to Tabs
- SublimeLinter: Show All Errors
- SublimeLinter: Show Line Errors
- Build: New Build System
- Vintageous: Toggle Vim Ctrl Keys

The screenshot shows the Sublime Text interface with two tabs open: 'first.py' and 'functions.py'. A command palette is open on the right side, displaying a list of packages and commands. The list includes:

- A Xiki Clone for Sublime Text 2/3
- install v2015.05.26.06.22.27; github.com/swdunlop/antiki
- ANTLR syntax highlight
- Antlr syntax highlighting for Sublime Text 2/3
- install v2015.01.05.08.51.51; github.com/iulius/SublimeText2-Antlr-syntax
- AoE2 RMS Syntax Highlighting
- Age of Empires 2 Random Map Script syntax highlighting for Sublime Text
- install v0.2.2; packagecontrol.io/packages/AoE2%20RMS%20Syntax%20Highlighting
- AoiConsolePanelStartup
- A Sublime Text plugin to always show console panel at startup.
- install v0.1.0; github.com/AoiKuiyuyou/AoiConsolePanelStartup-SublimeText
- Apache Hive
- Hive support for Sublime Text (2/3)
- install v2016.09.21.08.22.26; github.com/glinmac/hive-sublime-text

These packages are installed in the **Packages->User** directory which includes the entire necessary configuration. To browse the packages, select **Preferences -> Browse Packages** option, as shown in the screenshot below:





These files are the customized configuration which can be changed as and when needed. The packages are created in JSON format.

Consider the **sublime-keymap** package created which includes following code:

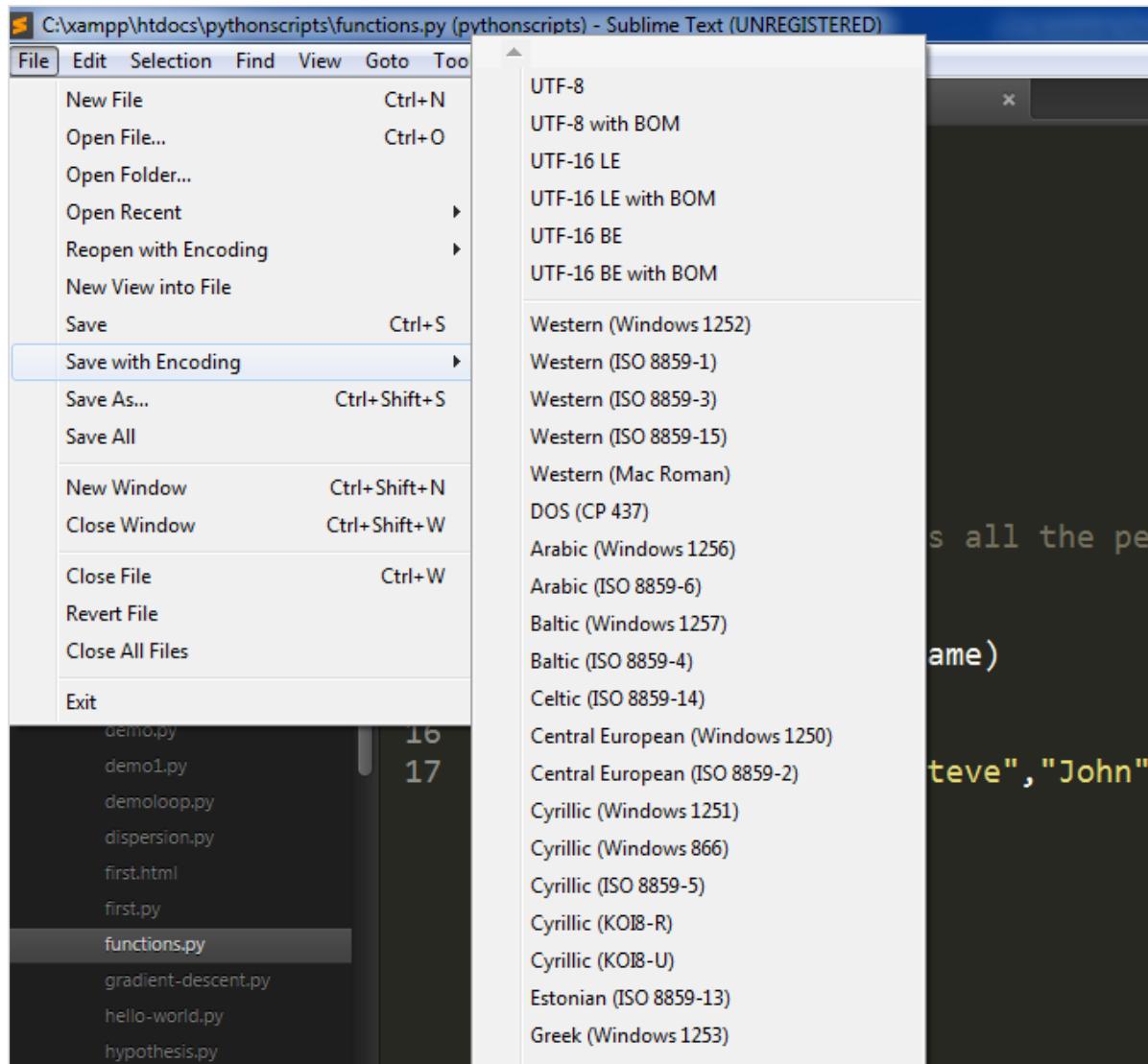
```
[{ "keys": ["super+alt+;"],  
  "command": "run_macro_file",  
  "args": {"file": "Packages/User/semicolon.sublime-macro"}  
}]
```

22. Sublime Text – Menus

In this chapter, you will learn about the primary options of **File**, **Edit** and **Goto** menus of Sublime Text editor.

Save with Encoding

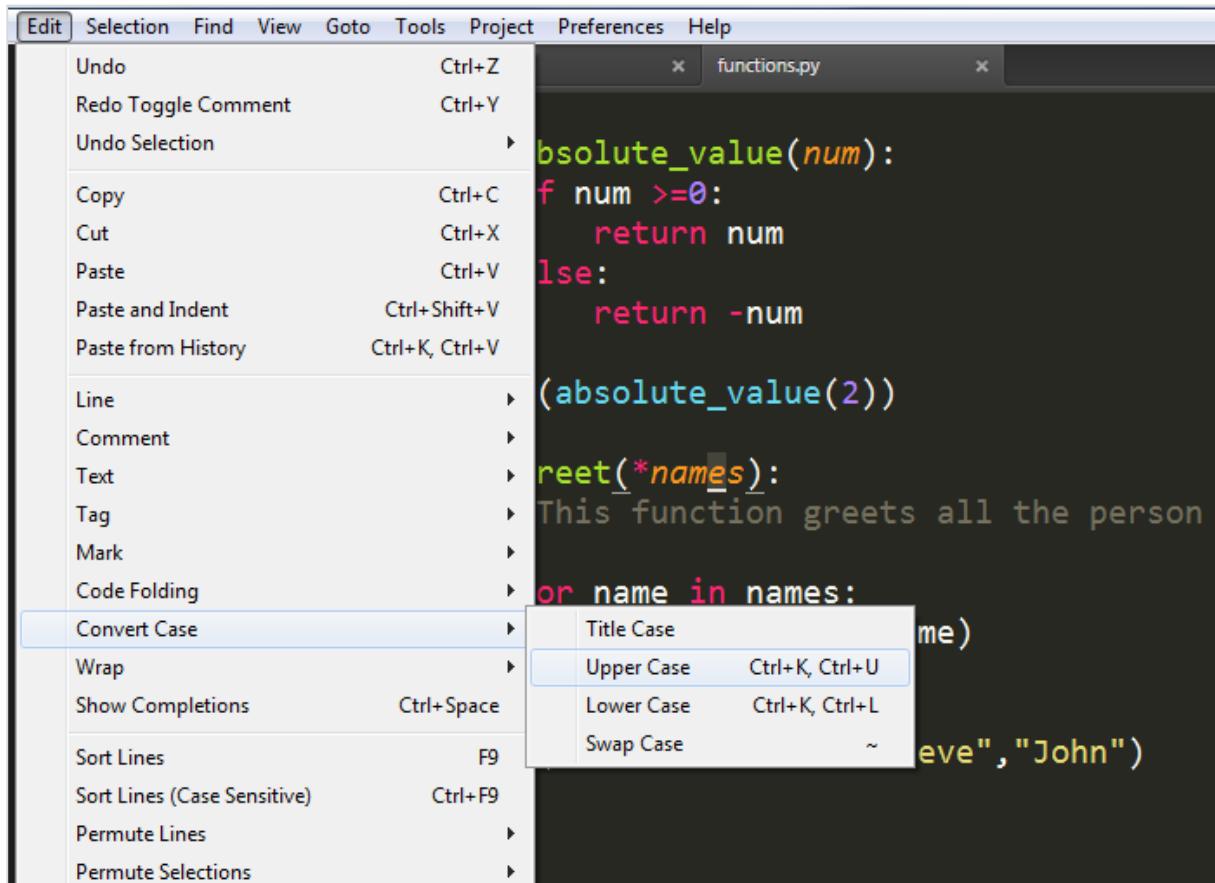
This option is included in the **File** menu which helps in saving the scripts and file with appropriate encoding. Sublime Text Editor includes a variety of options for encoding Python scripts, as shown in the following screenshot:



Encoding process helps the scripts to be secured from third-party attacks and allows the perceived item of use or interest which to be converted into a construct.

Convert Case

Convert case helps in the conversion of upper case to lower case and vice versa. Refer to the following screenshot for a better understanding:



For example, consider a keyword **Monica** included in the file and let us convert it to uppercase. Refer to the following screenshot for a better understanding:

```

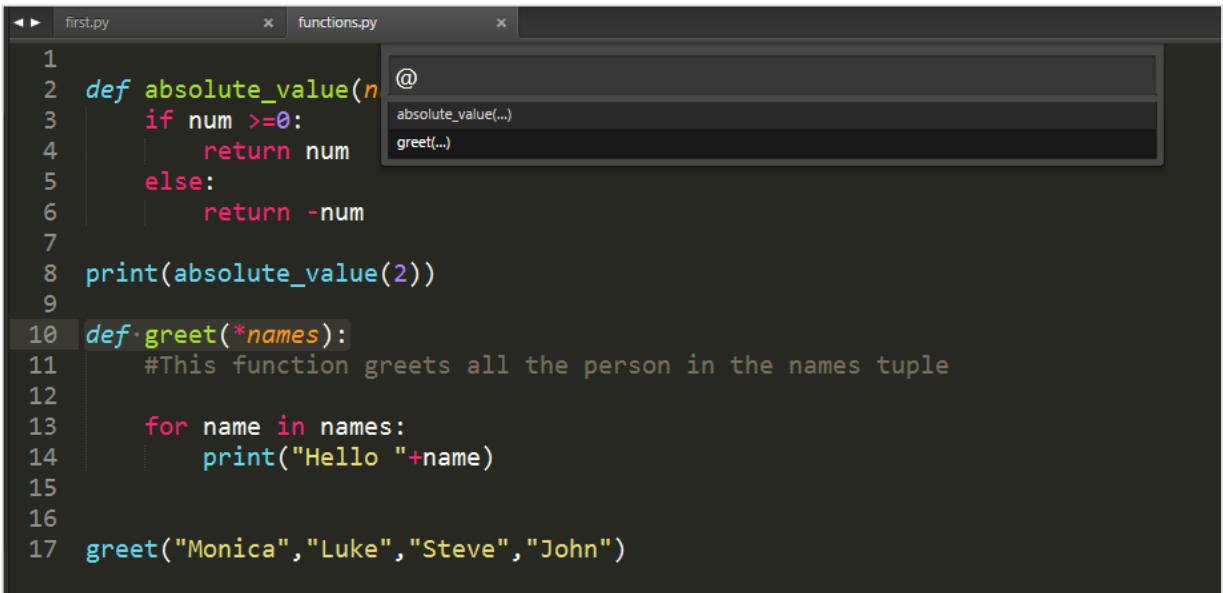
13     for name in names:
14         print("Hello "+name)
15
16
17 greet_("MONICA","Luke","Steve","John")

```

Goto Symbol

With this option, a user can search or get the appropriate keyword or any function. This option is very beneficial when we have more than 1000 lines of code and user got to search a unique line or annotation.

The following screenshot shows the demonstration of Goto Symbol:

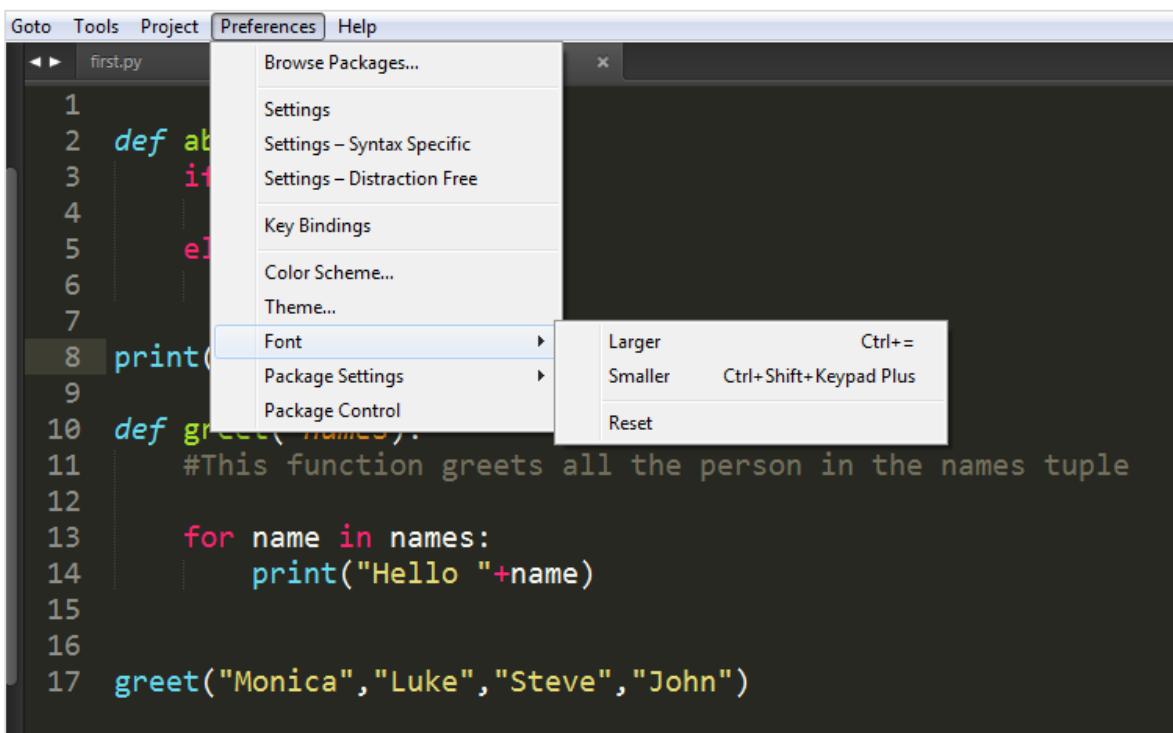


The screenshot shows a Sublime Text interface with two tabs: "first.py" and "functions.py". The "first.py" tab is active and contains the following Python code:`1 def absolute_value(n)
2 if num >=0:
3 return num
4 else:
5 return -num
6
7
8 print(absolute_value(2))
9
10 def greet(*names):
11 #This function greets all the person in the names tuple
12
13 for name in names:
14 print("Hello "+name)
15
16
17 greet("Monica","Luke","Steve","John")`A tooltip window is open over the word "greet" in line 10, showing the definition of the function. The tooltip title is "@", and it lists "absolute_value(...)" and "greet(...)".

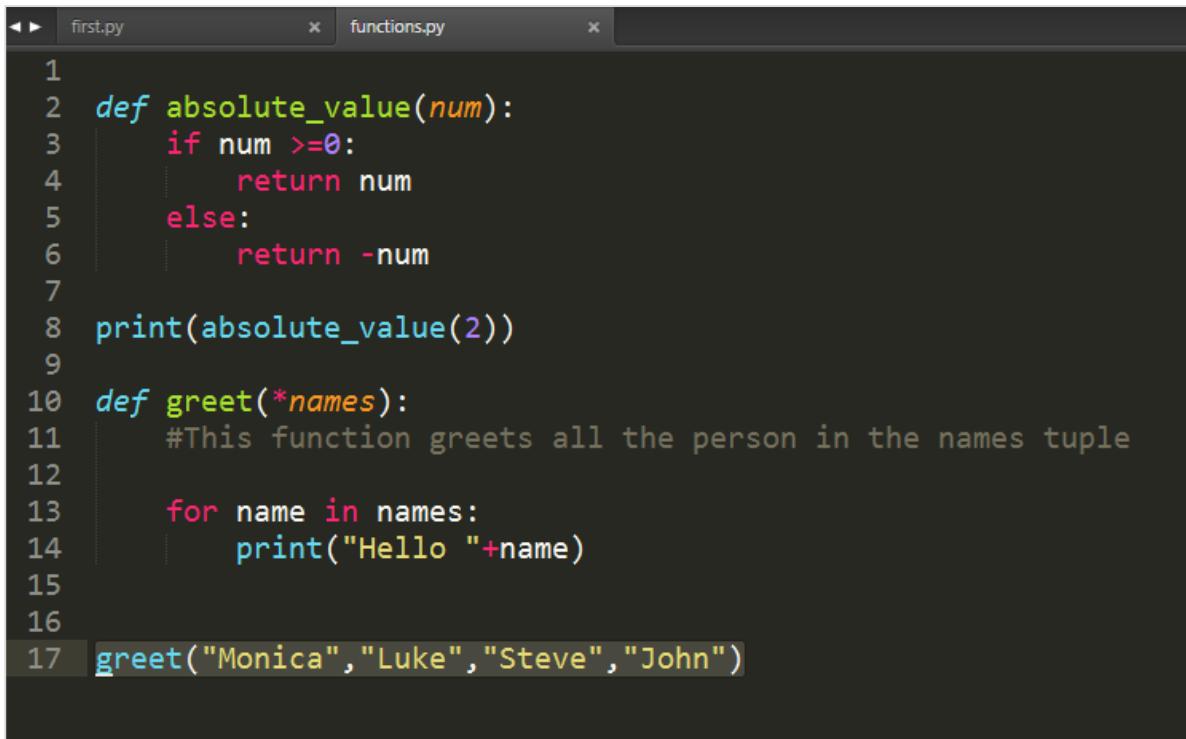
23. Sublime Text – Sub Menus of Font

The previous chapters discussed various options related to the **Preferences** menu. In this chapter, you will learn in detail about the **Font settings** of Sublime Text editor.

Sublime Editor offers three dimensions of font sizes: **Larger**, **Smaller** and **Reset** option to undo the particular changes made. Implementing the specified font is possible once the user selects the appropriate paragraph or text in editor.



Consider a selected text in the file **functions.py** where you need to make desired changes:

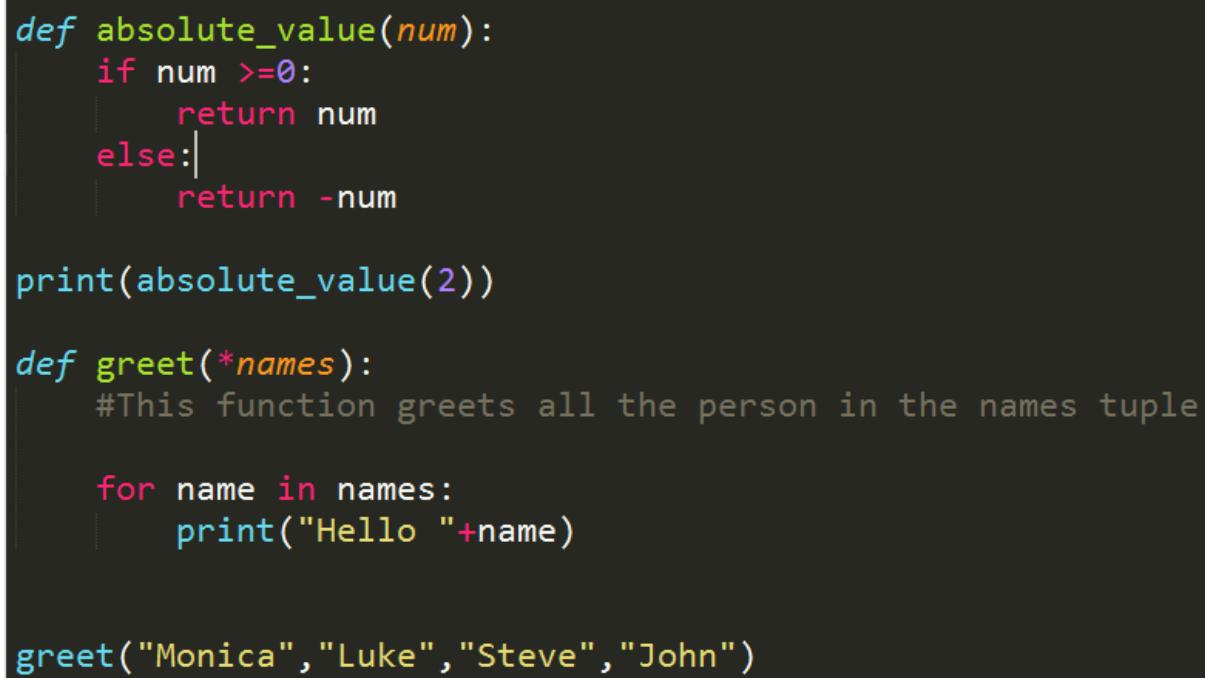


```

1
2 def absolute_value(num):
3     if num >=0:
4         return num
5     else:
6         return -num
7
8 print(absolute_value(2))
9
10 def greet(*names):
11     #This function greets all the person in the names tuple
12
13     for name in names:
14         print("Hello "+name)
15
16
17 greet("Monica", "Luke", "Steve", "John")

```

Observe that after the implementation of larger font, the dimensions and font size of the scripts will be automatically changed to larger dimension.



```

def absolute_value(num):
    if num >=0:
        return num
    else:
        return -num

print(absolute_value(2))

def greet(*names):
    #This function greets all the person in the names tuple

    for name in names:
        print("Hello "+name)

greet("Monica", "Luke", "Steve", "John")

```

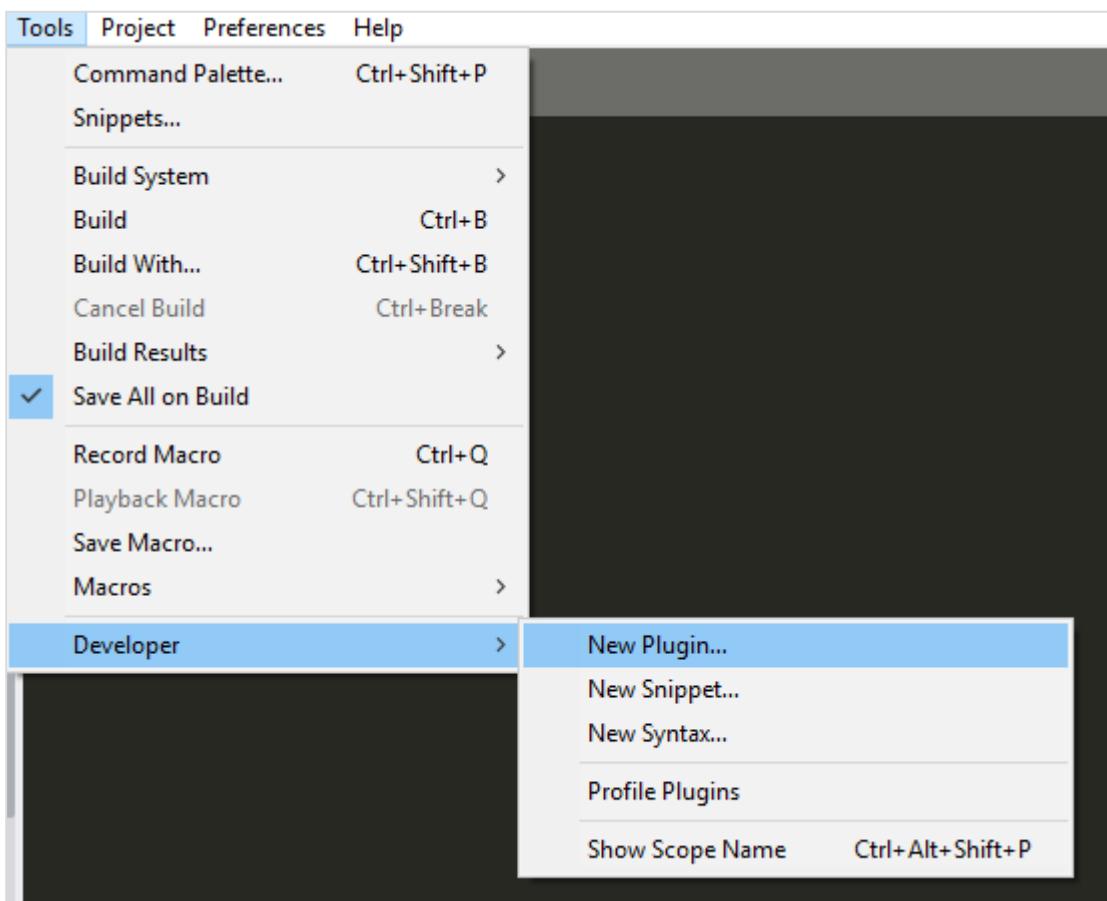
24. Sublime Text – Developing Plugin

Every editor includes plugin for the development, that triggers set of activities and default packages. Sublime Text editor includes a feature for developing your own customized plugin. This chapter discusses in detail about developing your own plugin in Sublime Text.

Developing Plugin

The following steps show you in detail how to develop a plugin in Sublime Text:

Step 1: Select the **New Plugin** option by navigating through **Tools -> Developer -> New Plugin** as shown below:



Step 2: The basic code of a plugin includes import of two main libraries: **sublime** and **sublime_plugin**.

```
import sublime
import sublime_plugin

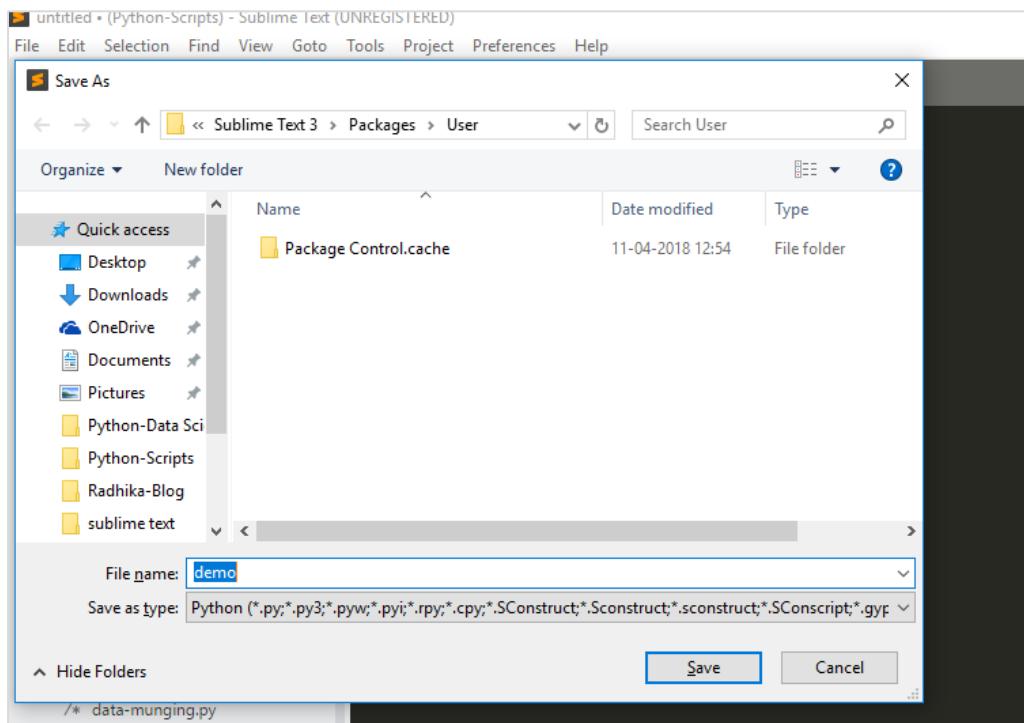
class ExampleCommand(sublime_plugin.TextCommand):
    def run(self, edit):
        self.view.insert(edit, 0, "Hello, World!")
```

The code for the plugin is:

```
import sublime
import sublime_plugin

class ExampleCommand(sublime_plugin.TextCommand):
    def run(self, edit):
        self.view.insert(edit, 0, "Hello, World!")
```

Step 3: The customized plugins are saved in **Packages->User** folder. Refer to the following screenshot that gives you the complete understanding of the plugins saved in Sublime Text editor.



Running the plugin

When you have created a plugin and saved it, open the console using the shortcut key **Ctrl+`** on Windows and **Cmd+`** on OSX, and execute the command shown here:

```
view.run_command(plugin-name)
```

```
C:\Users\DELL\Desktop\Python-Scripts\first.py (Python-Scripts) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
```

```
first.py x demo.py x
```

```
1 for i in [1,2,3,4,5]:
2     print (i)
```

```
/* activity2.py
/* activity3.py
/* activity4.py
/* activity5.py
/* activity6.py
/* activity7.py
/* activity8.py
Advertising.csv
/* bar-chart1.py
/* bar-chart2.py
bar.png
/* barchart-png.py
barcharts.pdf
Big Mart Dataset.xlsx
/* bubble-chart.py
/* case-study.py
/* central-limit.py
/* central-tendencies.py
chart.py
/* contour-plot.py
/* create-png.py
/* data-munging.py
/* demo.py
/* demoloop.py
```

```
Running python -u "C:\Users\DELL\Desktop\Python-Scripts\first.py"
Running python -m py_compile "C:\Users\DELL\Desktop\Python-Scripts\first.py"
: Info: processing 'py_compile': please wait...
reloading plugin User.demo
Could not import subprocess32 module, falling back to subprocess module
```

```
view.run_command('demo')
```

```
Line 2 Column 14
```

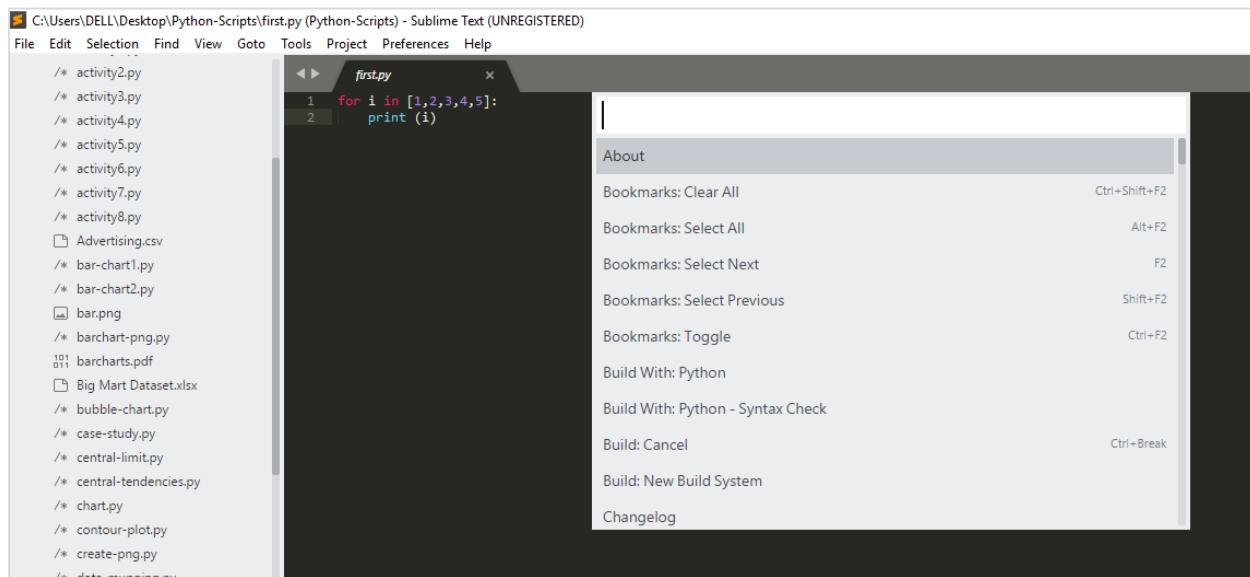
This command will execute the plugin defined by the user with the list of activities included in it.

25. Sublime Text – Command Palette

Command Palette includes a list of items or commands which are used frequently. The entries of commands are included in the **.sublime-commands** file.

Using Command Palette

To open a command palette in Sublime Text editor, you can use the shortcut key combination **Ctrl+Shift+P** on Windows and **Cmd+Shift+P** on OSX.

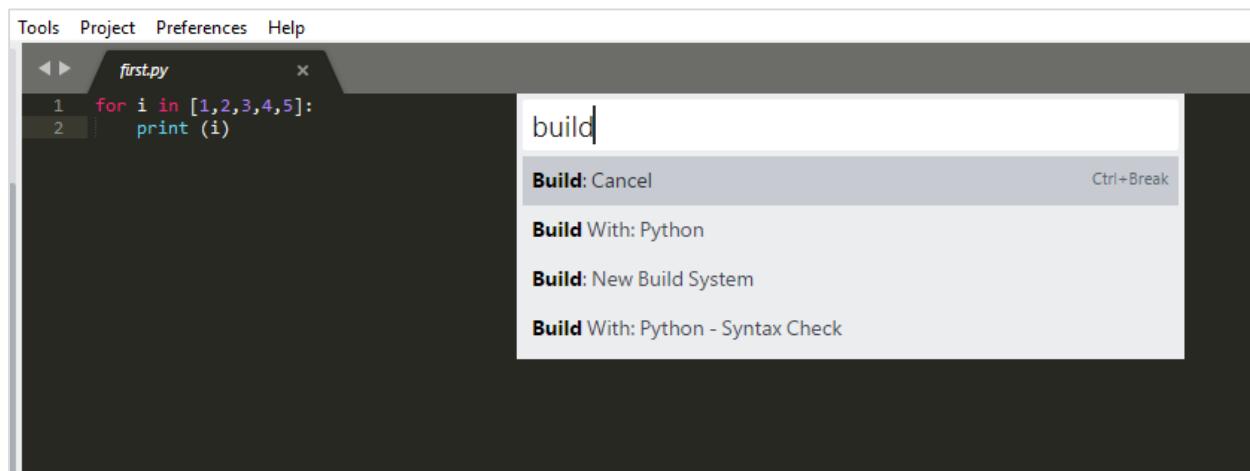


The commonly used commands from the palette are:

- Build with Python
- Install Packages

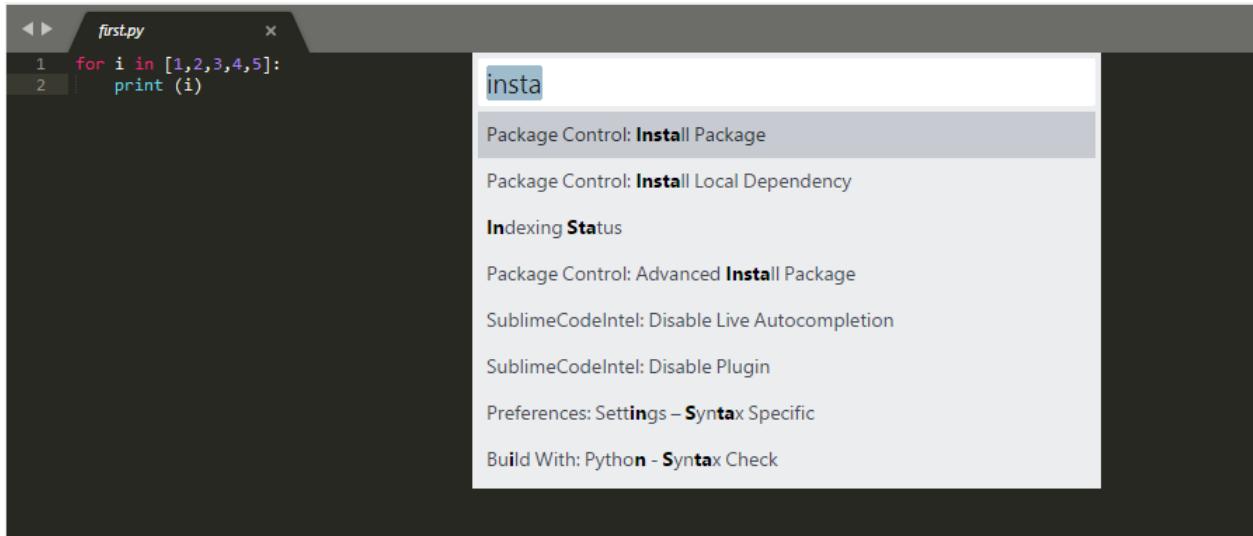
Build with Python

This will generate all the dependencies and build the specified code of a given Python file.



Install Packages

With this command, we get list of packages which can be installed which is not included earlier.



All the commands included in the command palette are stored in the **Packages** directory. The basic example of command declared inside the **Default.sublime-commands** file is shown in the code given below:

```
[  
  { "caption": "Project: Save As", "command": "save_project_as" },  
  { "caption": "Project: Close", "command": "close_project" },  
  { "caption": "Project: Add Folder", "command": "prompt_add_folder" },  
]
```

Note:

The JSON file includes 3 main keys for every command:

- Name/Caption
- Location
- Content

26. Sublime Text – Debugging PHP Application

Debugging is the process of finding errors and bugs in the given code and fixing them. Sublime editor includes various plugins that have debugging features, which helps in finding errors easily.

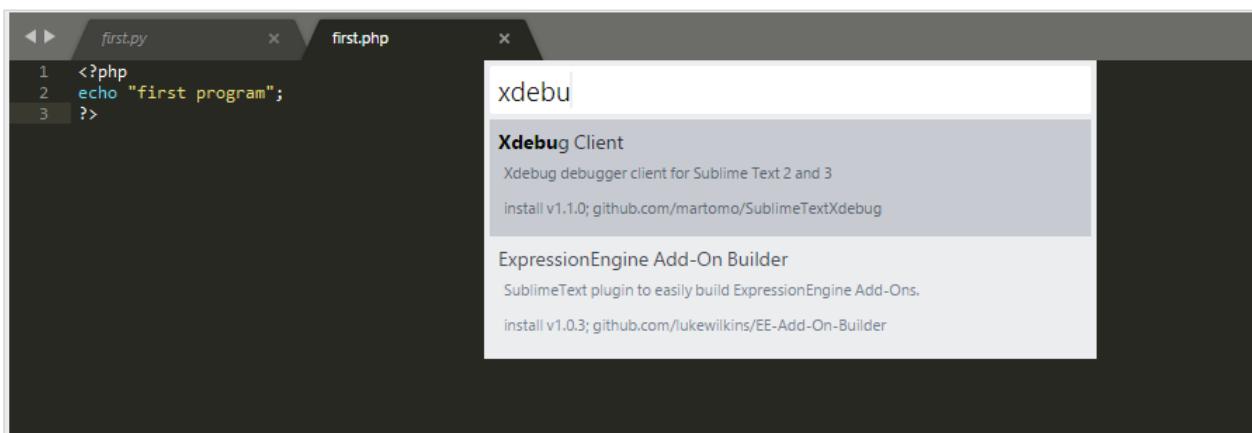
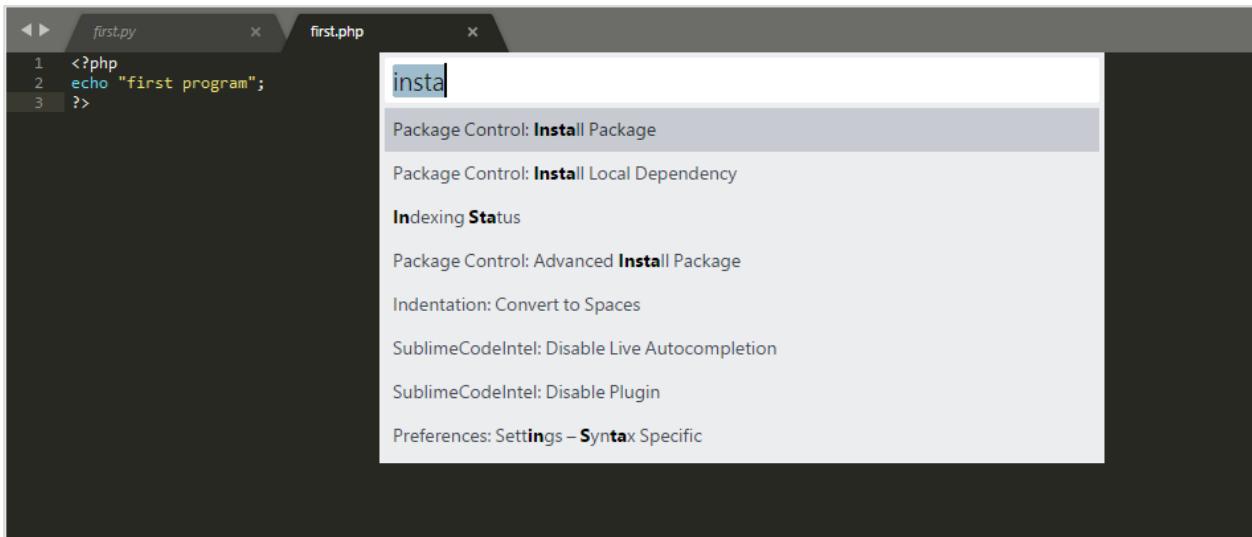
In this chapter, you will learn how to debug a PHP web application. Sublime uses **Sublime TestX debug** plugin for this purpose. The features of this plugin are:

- It is an extension used for debugging the PHP files and scripts.
- Provides a list of debugging and profiling capabilities.
- Maintains stack traces and real time parameters like display and functionalities.

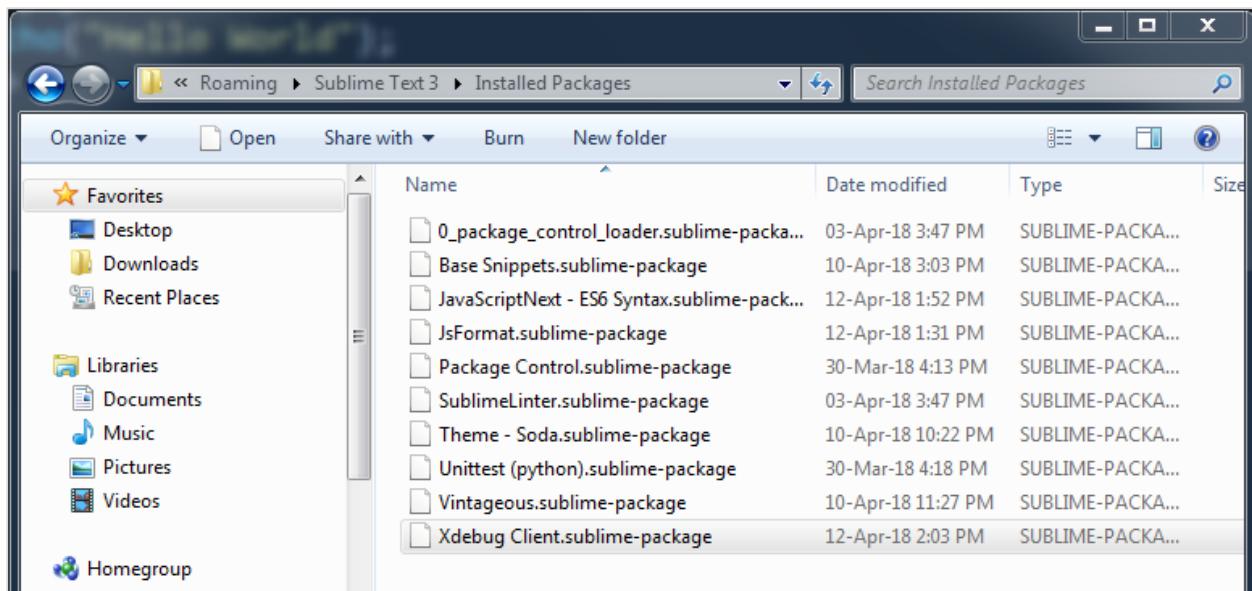
Installation of Sublime Plugin

To install Sublime Text Xdebug plugin, you will have to take the following steps:

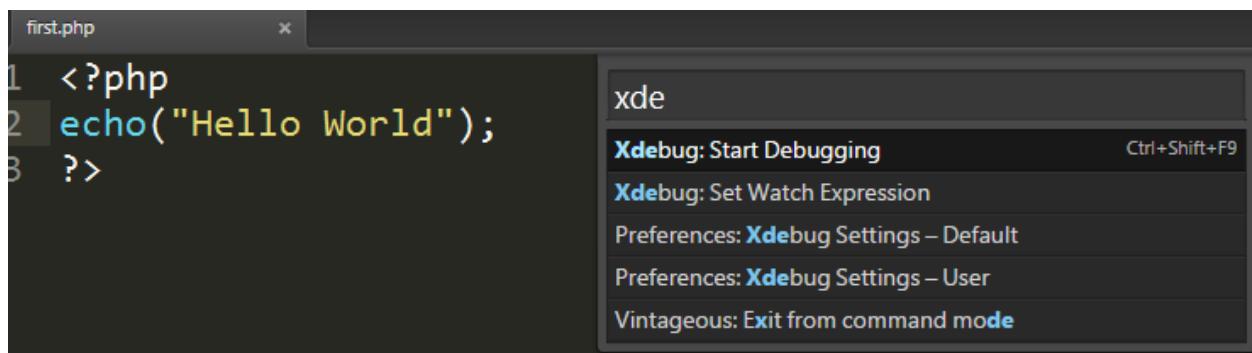
Step 1: Install package control palette installation with the shortcut of **Ctrl+Shift+P** or **Cmd+shift+P**.



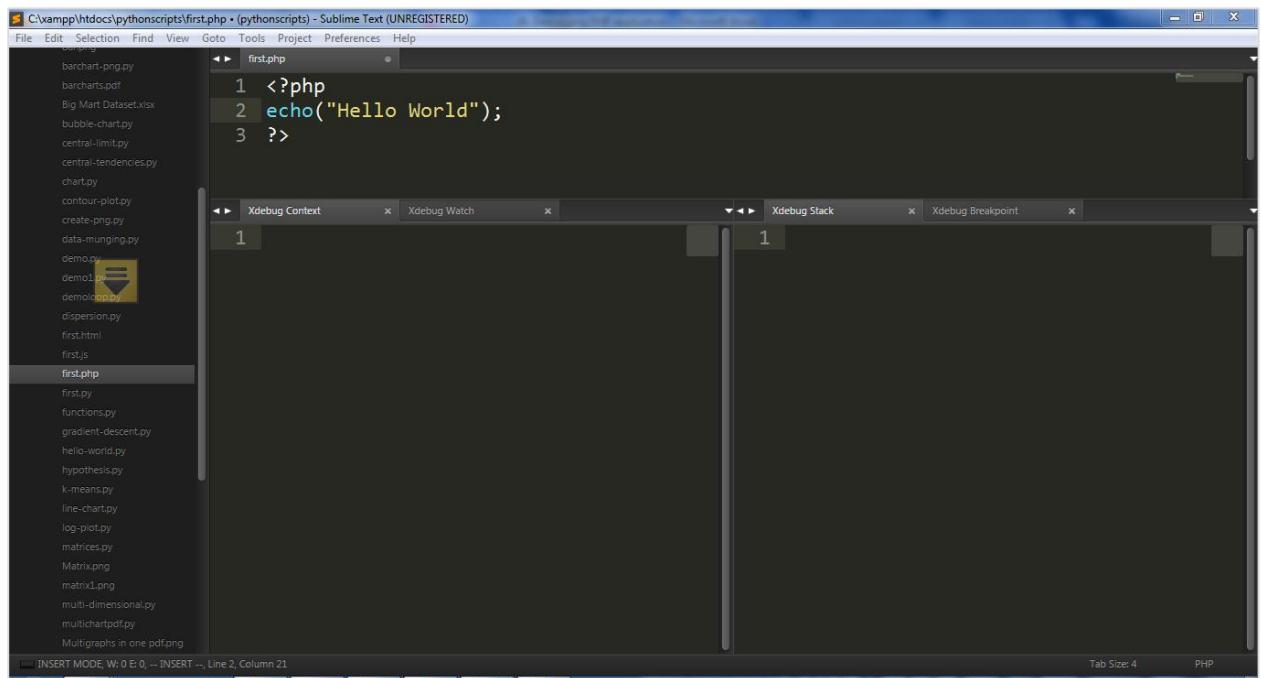
Step 2: After the successful installation of Xdebug client, the configuration file will be kept in the **Installed Packages** folder. The configuration file is the JSON file which includes the parameters for debugging a PHP web application.



Step 3: Start the debug session of PHP file using the shortcut **Ctrl+F9**. You can also do it using the **Start Debug** option using command palette.



Step 4: The split windows of **XdebugClient** will show the output of PHP file. It keeps a process of debugging code line by line. Refer to the following screenshot for a better understanding:



The screenshot shows the Sublime Text editor interface with the title bar "C:\xampp\htdocs\pythonscripts\first.php • (pythonscripts) - Sublime Text (UNREGISTERED)". The left sidebar lists various files in the project. The main editor window displays the following PHP code:

```
1 <?php
2 echo("Hello World");
3 ?>
```

Below the editor, there are four tabs labeled "Xdebug Context", "Xdebug Watch", "Xdebug Stack", and "Xdebug Breakpoint". The status bar at the bottom indicates "INSERT MODE, W: 0 E: 0 -- INSERT --, Line 2, Column 21" and "Tab Size: 4 PHP".

27. Sublime Text – Debugging JavaScript Application

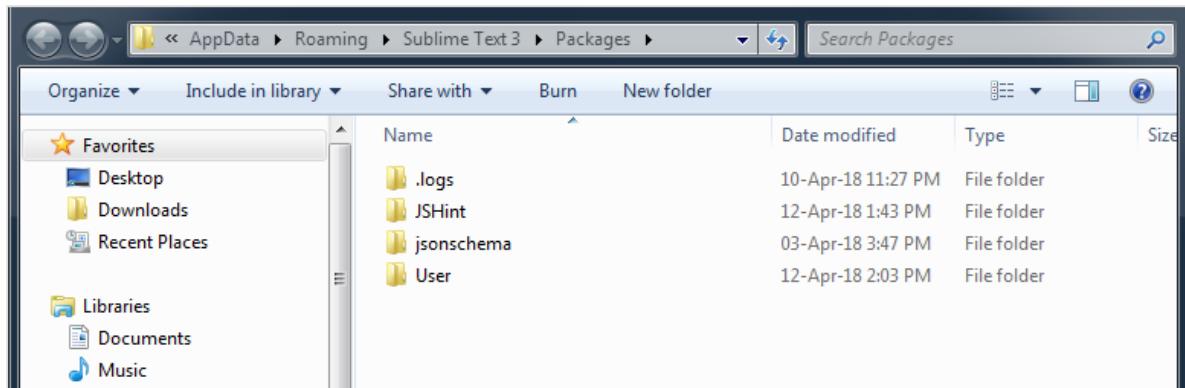
Sublime Text Editor has a plugin **Web Inspector** for debugging JavaScript code. It helps us to set breakpoints, examine the console and evaluate sections of code. In this chapter, you will learn in detail about this plugin.

Installation of Sublime Web Inspector

The following steps show the installation of Sublime Web Inspector:

Step 1

Use the menu option **Preferences -> Browse Packages** to get the path of installed packages as shown in the given screenshot.



Step 2

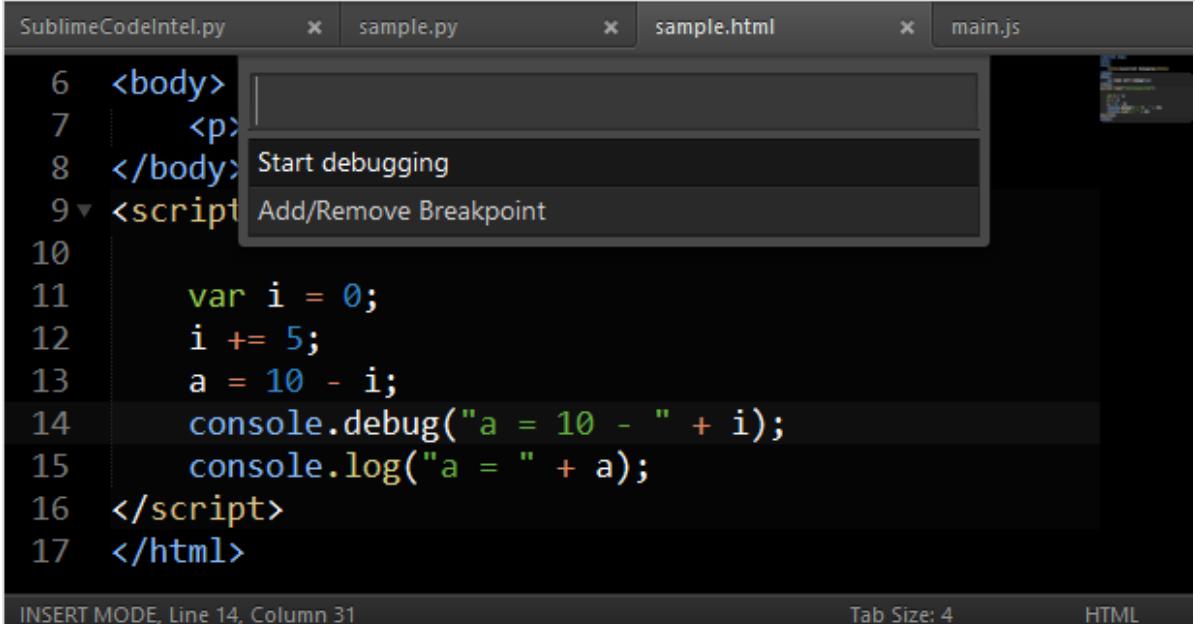
With the help of **git** command prompt, clone the packages of Debugging Javascript web application using the git repository, as shown in the command given below:

```
git clone -b ST3 "git://github.com/sokolovstas/SublimeWebInspector.git"
```

A screenshot of a terminal window titled 'Git CMD'. The command 'git clone -b ST3 "git://github.com/sokolovstas/SublimeWebInspector.git"' is entered. The output shows the cloning process: 'Cloning into 'SublimeWebInspector'...', 'remote: Counting objects: 1716, done.', 'Receiving objects: 100% (1716/1716), 1005.27 KiB | 134.00 KiB/s, done.', 'remote: Total 1716 (delta 0), reused 0 (delta 0), pack-reused 1716', 'Resolving deltas: 100% (910/910), done.', 'Checking connectivity... done.' The prompt 'C:\Users\Admin>' is visible at the bottom.

Step 3

To test the installation, open a command palette and start the web inspector. You can use the shortcut key for web inspector **Ctrl+Shift+R** on Windows and **Cmd+Shift+R** on OSX systems. You can see all the breakpoints needed for JavaScript application, as shown below:



The screenshot shows the Sublime Text interface with several tabs: SublimeCodeIntel.py, sample.py, sample.html, and main.js. The main.js tab contains the following code:

```
6 <body>
7   <p>
8 </body>
9 <script> Start debugging
10
11   var i = 0;
12   i += 5;
13   a = 10 - i;
14   console.debug("a = 10 - " + i);
15   console.log("a = " + a);
16 </script>
17 </html>
```

A context menu is open over the line containing the script tag, with the option "Add/Remove Breakpoint" highlighted.

Bottom status bar: INSERT MODE, Line 14, Column 31 | Tab Size: 4 | HTML

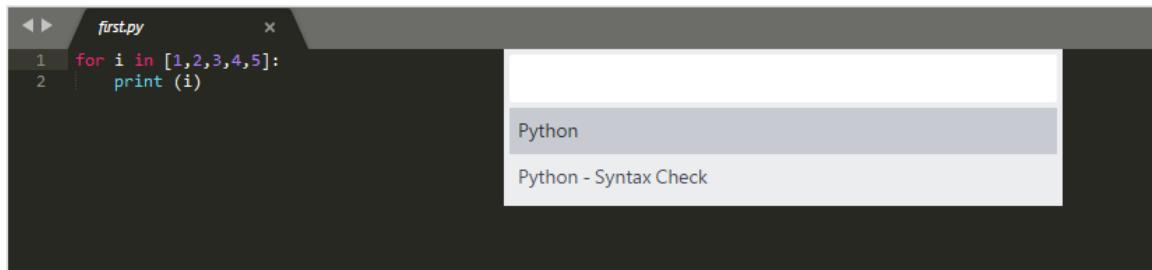
28. Sublime Text – Batch Processing

Batch processing in Sublime Text also refers to **Build systems**. Build systems helps user to run files through external programs such as **make**, **tidy** and **interpreters**.

The following points are worth notable while working with Build systems:

- They are JSON files and have the extension **.sublime-build**.
- To initiate the build process, you can select the option **Tools -> Build** or the shortcut **Ctrl+B** on Windows and **Cmd+B** for OSX system.

Recall that throughout this tutorial, we are focusing on Python files. Sublime text includes two build modes for python files.

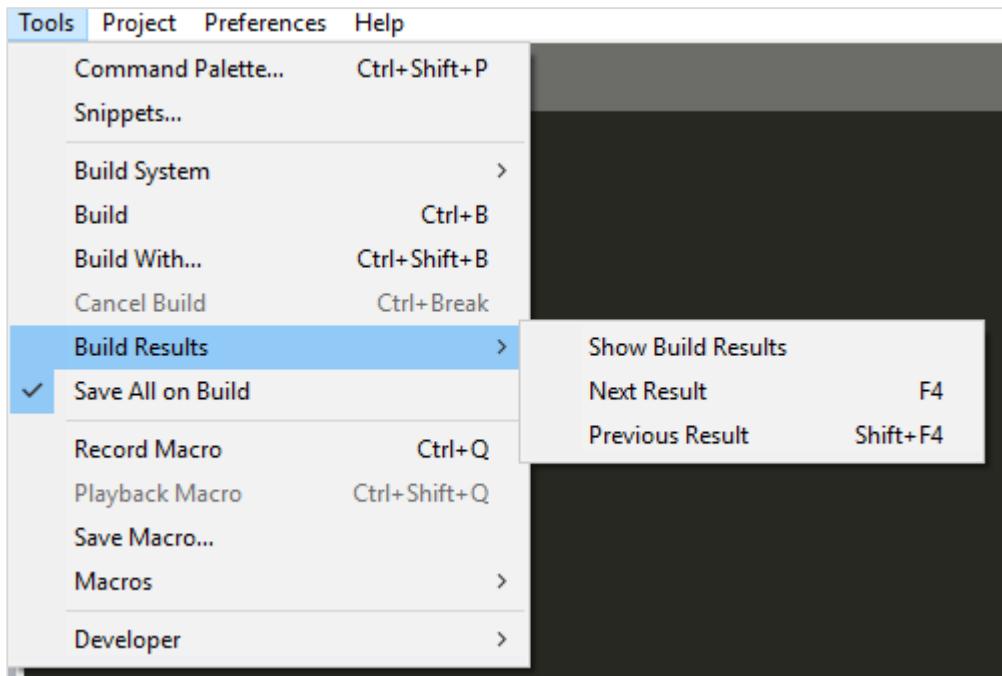


Once the build for Python is completed, you can observe the following output on the editor:

```
for i in [1,2,3,4,5]:
    print (i)
```

```
1
2
3
4
5
[Finished in 6.5s]
```

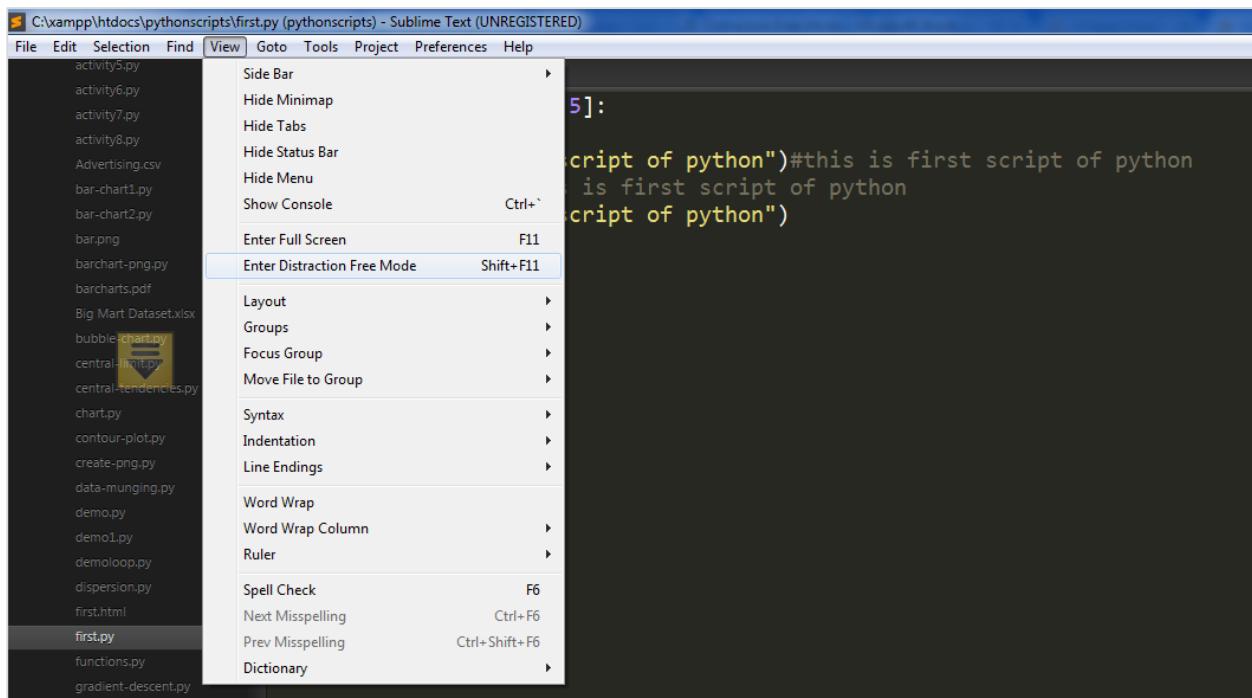
You can see the list of build results created for the specific project in Sublime Text editor using the option **Tools -> Build Results-> Show Build Results**.



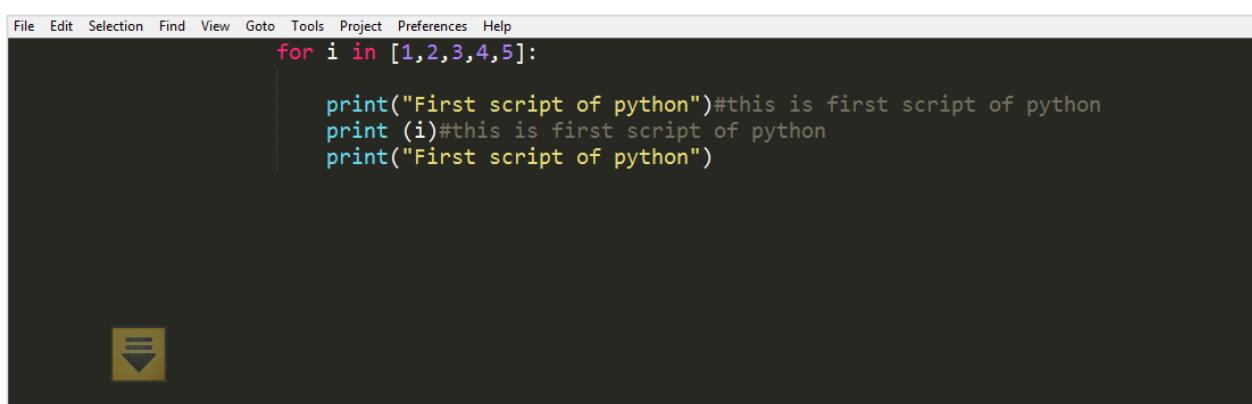
Note that build systems and the complete files associated with batch processing must be located under the **Packages** folder (**Packages/user**). Many packages in Sublime Editor include their own build systems.

29. Sublime Text – Distraction Free Mode

Distraction Free Mode in Sublime Text editor allows displaying the code and files full screen. You can enter distraction free mode via option **View -> Enter distraction mode** menu item.



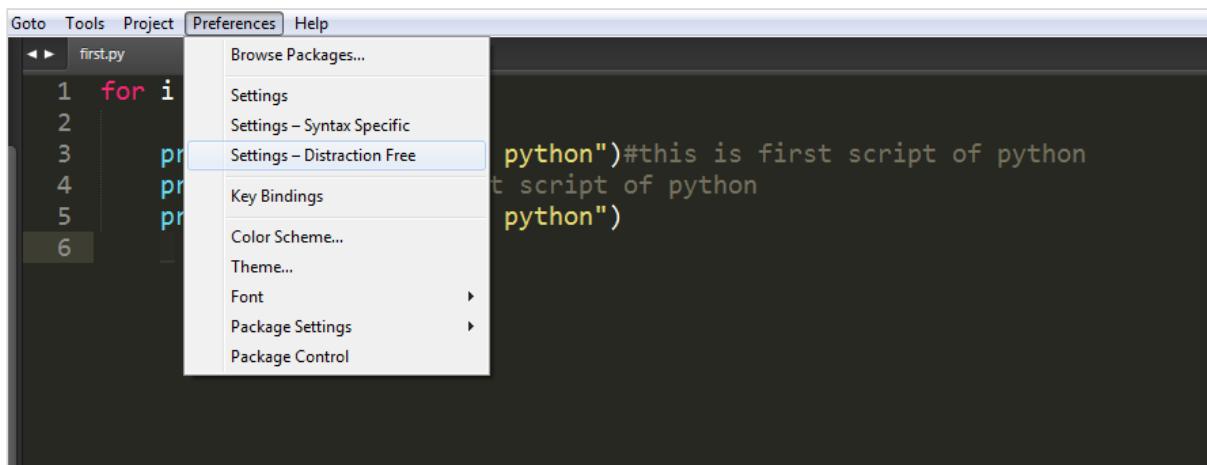
Once the distraction mode is enabled in the Sublime Text editor, the output will be visible as below:



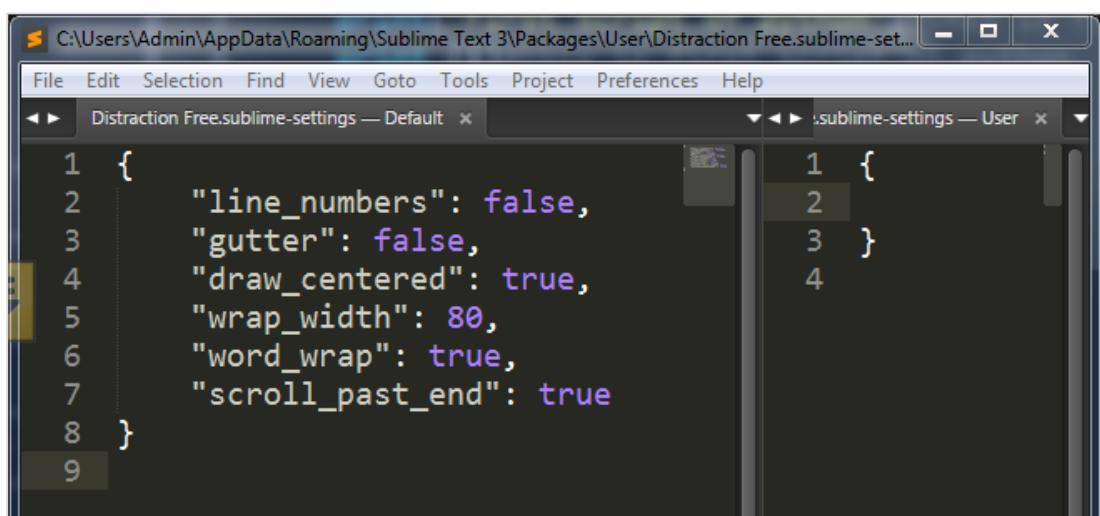
All UI chrome accessible items are hidden, but accessible in distraction mode.

Customization

All the properties of customization of distraction free mode of Sublime Text editor are included in **Distraction Free.sublime-settings** JSON file as shown in the screenshot given below.



The following screenshot shows the properties included in Distraction free mode of Sublime editor:



Note that the attributes like **line_numbers**, **gutter**, **draw_centered**, **wrap_width**, **word_wrap** and **scroll_past_end** are such that they enable attractive distraction free mode.

30. Sublime Text – SublimeCodeIntel Plugin

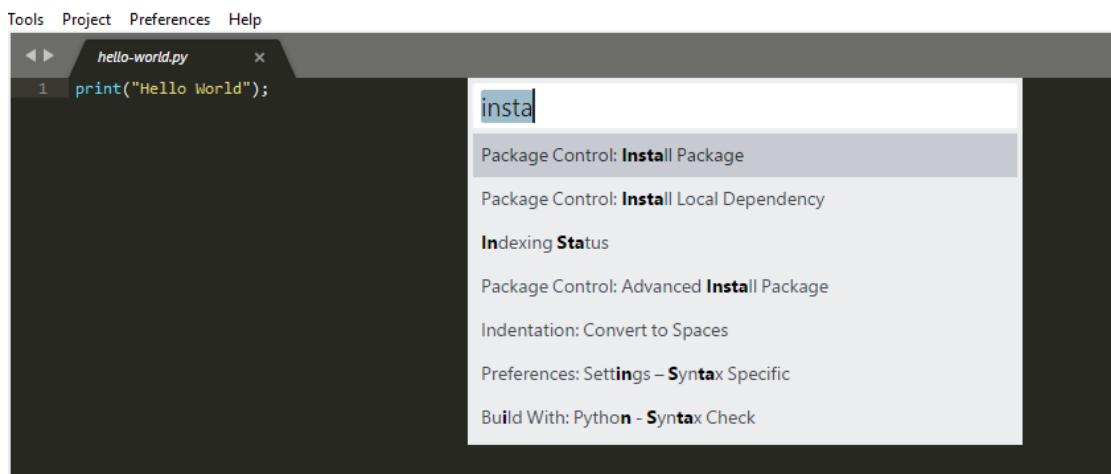
SublimeCodeIntel is an important plugin for the Sublime Text editor. It offers the following features:

- Jump to symbol feature, which helps the user to jump to the file and the specified line of the symbol.
- Includes library of autocomplete and displays the modules/symbols in real time.
- Tooltips display information in the status bar about the associated function.

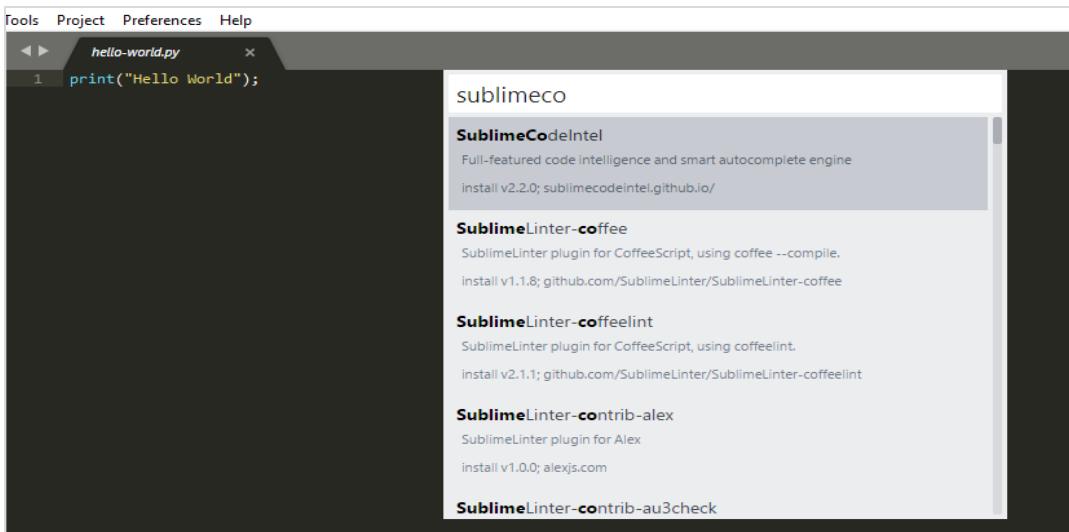
Installation

You can install the SublimeCodeIntel plugin through the following steps:

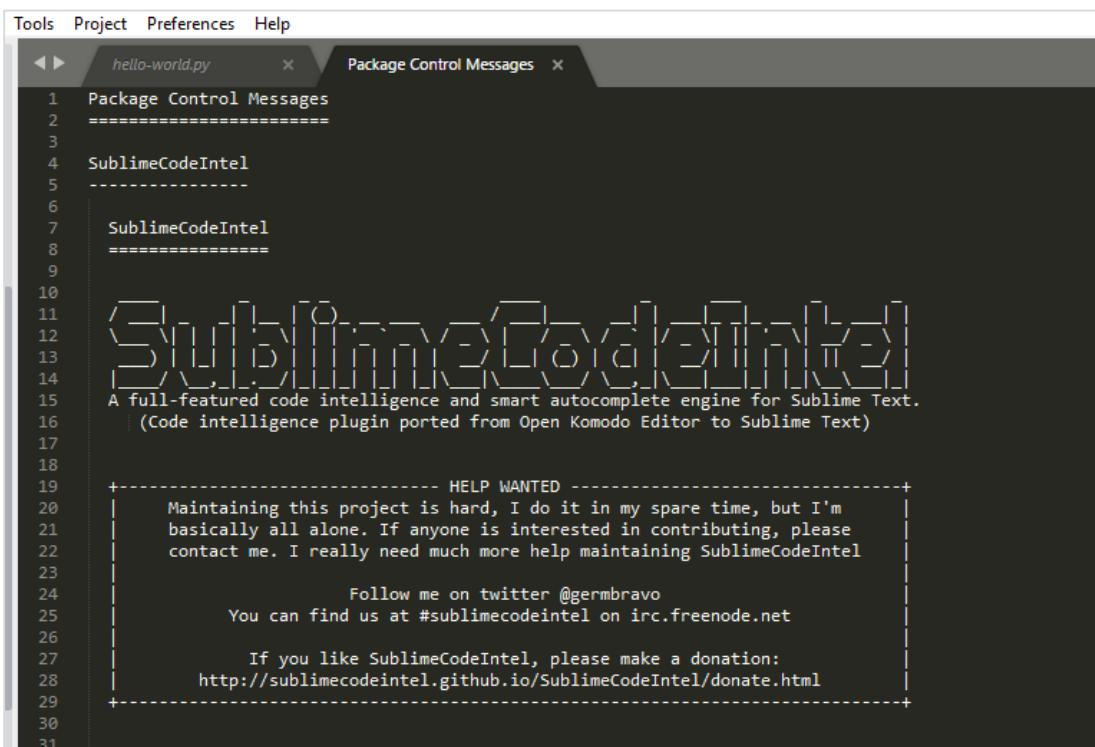
Step 1: Get the install package functionality of Sublime Text editor using the shortcut **Ctrl+Shift+P** on Windows and **Cmd+Shift+P** for OSX system. Refer to the following screenshot for better understanding:



Step 2: Now, select the plugin **SublimeCodeIntel** for installation and restart the Sublime Text editor.



Step 3: On successful installation of **SublimeCodeIntel** plugin, your screen will look like this:

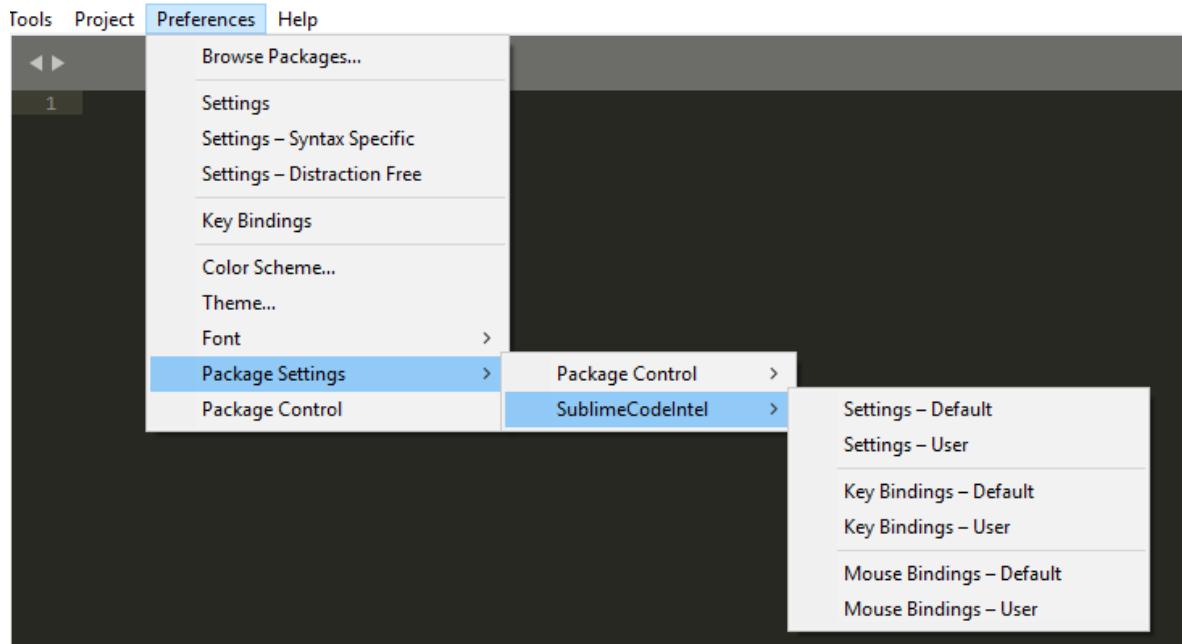


Configuration

There are two in built configurations of SublimeCodeIntel:

- Settings – Default
- Settings –User

Refer to the following screenshot that shows these configurations:



The configuration for **SublimeCodeIntel** is included in the JSON file called **SublimeCodeIntel.sublime-settings**. The configuration needed for each language is included in SublimeCodeIntel plugin as below:

```
/*
    Defines a configuration for each language.
*/
"codeintel_language_settings": {
    "Python3": {
        "python3": "/usr/local/bin/python3.3",
        "codeintel_scan_extra_dir": [
            "/Applications/Sublime Text.app/Contents/MacOS",
            "~/Library/Application Support/Sublime Text
3/Packages/SublimeCodeIntel/arch",
            "~/Library/Application Support/Sublime Text
3/Packages/SublimeCodeIntel/libs"
        ],
        "codeintel_scan_files_in_project": true,
        "codeintel_selected_catalogs": []
    }
}
```

```
},
"JavaScript": {
    "codeintel_scan_extra_dir": [],
    "codeintel_scan_exclude_dir": ["/build/", "/min/"],
    "codeintel_scan_files_in_project": false,
    "codeintel_max_recursive_dir_depth": 2,
    "codeintel_selected_catalogs": ["jQuery"]
},
"PHP": {
    "php": "/Applications/MAMP/bin/php/php5.5.3/bin/php",
    "codeintel_scan_extra_dir": [],
    "codeintel_scan_files_in_project": true,
    "codeintel_max_recursive_dir_depth": 15,
},
"codeintel_scan_exclude_dir": ["/Applications/MAMP/bin/php/php5.5.3/"]
}
```

These configurations can be customized as and when needed. It depends on the number of modules/libraries installed by the user with various parameters such as the size and complexity of the project a user is working on.