




Topic	PYTHON INSTALLATION	
Class Description	The student will set up their local environment to write and test code using Python. The student will be introduced to different modules of Python. The student will create a Stopwatch using the time module.	
Class	PRO C101	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> <li>• Install Python</li> <li>• Learning to create and run a code in VSC.</li> <li>• Install pip</li> <li>• Introduction to howdoi</li> <li>• Create a Stopwatch using a time module.</li> </ul>	
Resources Required	<ul style="list-style-type: none"> <li>• Teacher Resources:               <ul style="list-style-type: none"> <li>○ Laptop with internet connectivity</li> <li>○ Earphones with mic</li> <li>○ Notebook and pen</li> <li>○ Visual Studio Code</li> </ul> </li> <li>• Student Resources:               <ul style="list-style-type: none"> <li>○ Laptop with internet connectivity</li> <li>○ Earphones with mic</li> <li>○ Notebook and pen</li> <li>○ Visual Studio Code</li> </ul> </li> </ul>	
Class structure	Warm-Up Teacher-led Activity Student-led Activity Wrap-Up	05 mins 15 mins 20 mins 05 mins
WARM-UP SESSION - 5 mins		
Teacher Starts Slideshow 		

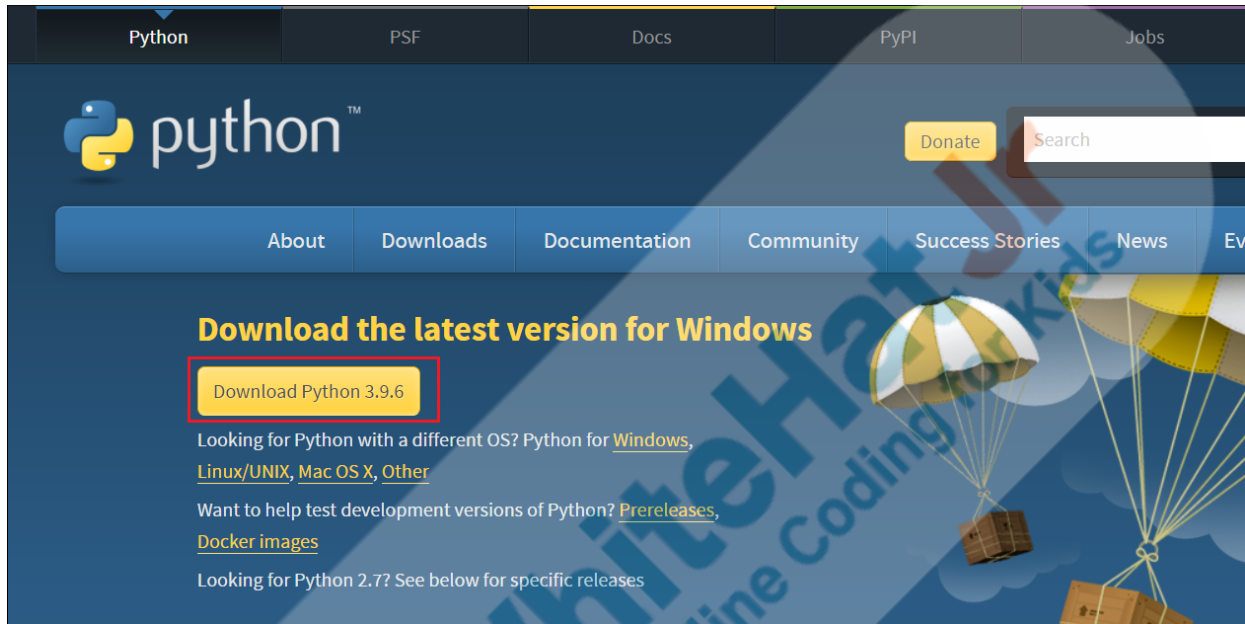
<p align="center"><b>Slide 1 to 3</b></p> <p align="center">Refer to speaker notes and follow the instructions on each slide.</p>	
<p>Hey &lt;student's name&gt;. How are you? It's great to see you! Are you excited to learn something new today?</p> <p><b>Following are the WARM-UP session deliverables:</b></p> <ul style="list-style-type: none"> <li>• Greet the student.</li> <li>• Revision of previous class activities.</li> <li>• Quizzes.</li> </ul>	<p><b>ESR:</b> Hi, thanks! Yes I am excited about it!</p> <p>Click on the slide show tab and present the slides</p>
<p align="center"><b>WARM-UP QUIZ</b> Click on In-Class Quiz</p>	
<p align="center"><b>Continue WARM-UP Session</b> Slide 4 to 14</p> 	
<p><b>Following are the session deliverables:</b></p> <ul style="list-style-type: none"> <li>• Appreciate the student.</li> <li>• Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.</li> </ul>	
<p align="center"><b>Teacher Action</b></p>	<p align="center"><b>Student Action</b></p>
<p>Did you try adding more records into Phone_Book?</p> <p><i><b>Note:</b> In case the student's reply is yes, you can check his work and appreciate it.</i></p>	<p><b>ESR:</b> Varied</p>
<p align="center"><b>Teacher Ends Slideshow</b></p> 	
<p align="center"><b>TEACHER-LED ACTIVITY 1- 15 mins</b></p>	
<p align="center"><b>Teacher Initiates Screen Share</b></p>	

<b>ACTIVITY</b> <ul style="list-style-type: none"> <li>• Install Python on the local system.</li> <li>• Introduce and install package manager.</li> </ul>	
Teacher Action	Student Action
<p>So far, we are using an online application Google Colab for writing and testing Python code.</p> <p>Today, we will install Python to our machine and will add a few extensions in VSC to run python code.</p> <p><b>Note 1:</b> The class includes installation steps for Windows and Mac as part of <a href="#">Teacher Activity 1</a>.</p> <p><b>Note 2:</b> This class also includes steps to prepare Visual Studio Code Editor to run the Python program in <a href="#">Teacher Activity 2</a></p> <p>There are various benefits of using local code editors vs. online code editors. I will quickly mention a few of them.</p> <p><b>Privacy:</b> You can't guarantee that no one ever had access to your documents in online services.</p> <p><b>Usability:</b> For online editors, continuous Internet connection is a must. For local editors, you can develop and run the code without the internet too.</p> <p><b>Compilation:</b> It is much faster while running code in local editors.</p> <p>On other hand, online editors greatly increase the collaborative editing of the documents.</p>	

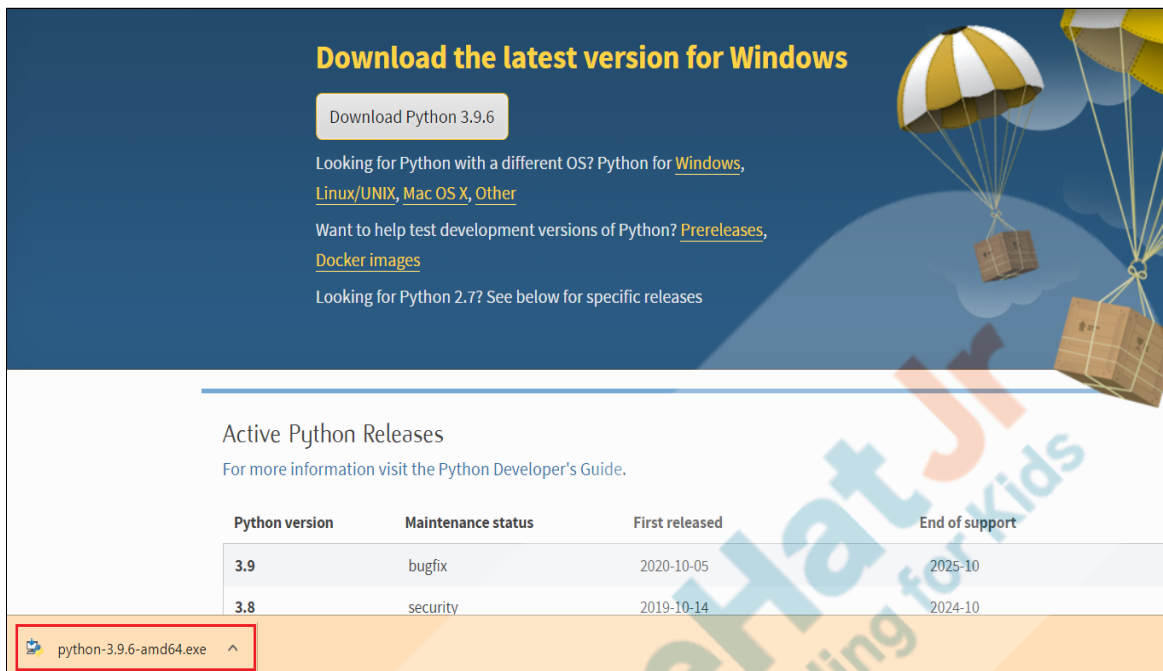
<p>Keeping these pros and cons in mind, let us start today's class.</p> <p>Are you ready to start?</p>	<p><b>ESR: Yes</b></p>
<p><b>Installing Python on Windows:</b></p> <ol style="list-style-type: none"> <li>1. Open <a href="https://www.python.org/downloads/">https://www.python.org/downloads/</a> to download.</li> </ol> <p><i><b>Note:</b> In case the teacher already has Python installed; Ask the student to share the screen while guiding him/her to install it. The teacher can ask the student to click on <a href="#">Student Activity 1</a> to download python for Windows/Mac.</i></p>	
	
<ol style="list-style-type: none"> <li>2. Click on Download Python 3.9.6 (<b>Note</b> that 3.9.6 denotes the version of the Python at the time of the creation of this document, this might vary for you.</li> </ol>	

Make sure to download the Python version 3 or higher.)

A file with the **.exe** extension will be downloaded.



*Note: A file with the **.exe** extension will be downloaded.*



**Download the latest version for Windows**

Download Python 3.9.6

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

Looking for Python 2.7? See below for specific releases

Active Python Releases

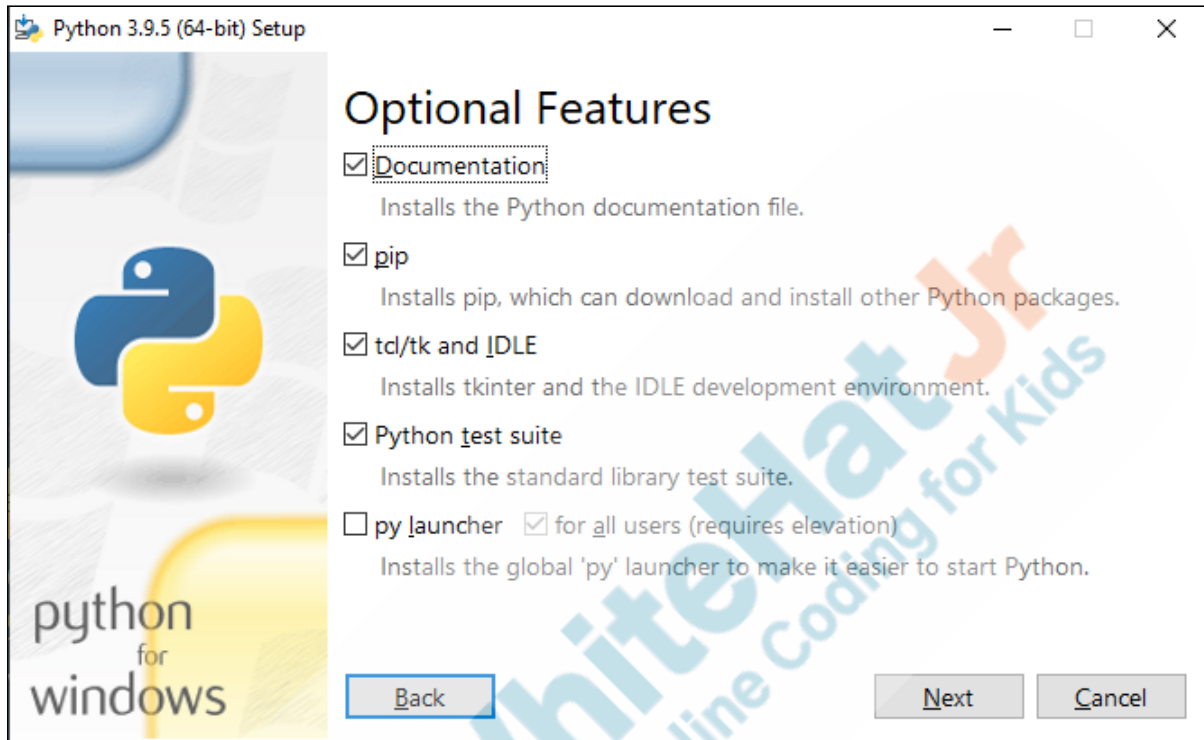
For more information visit the [Python Developer's Guide](#).

Python version	Maintenance status	First released	End of support
3.9	bugfix	2020-10-05	2025-10
3.8	security	2019-10-14	2024-10

python-3.9.6-amd64.exe ^

3. Open folder where the .exe file is downloaded and run the **.exe** file.
4. Select **"Install Now"**.
5. You will not need to be an administrator (you install the [Python Launcher for Windows](#) for all users).
6. Python will be installed into your user directory.
7. The [Python Launcher for Windows](#) will be installed according to the option at the bottom of the first page.
8. The standard library, test suite, launcher, and **pip** will be installed.
9. If selected, the install directory will be added to your PATH.
10. Shortcuts will only be visible for the current user.

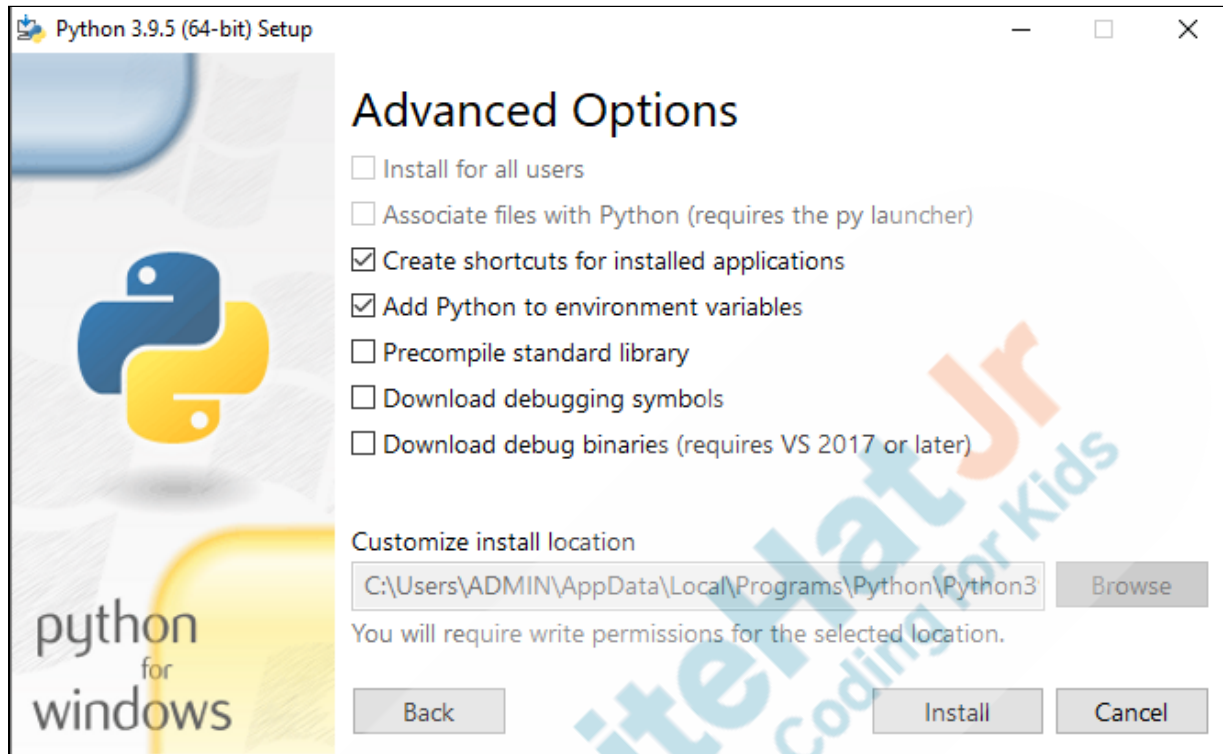
Check the boxes for the Optional features and click “**Next**”.



Note down the path where python is installed.

Check the boxes for:

11. “**Create shortcuts for installed applications**” and
12. “**Add Python to the environment variables**”
13. Click on Install.



**Note:** The installation speed depends on the system speed, usually taking 50-70 seconds.

### Installing Python on a Macintosh:

Mac OS X 10.8 comes with Python 2.7 pre-installed by Apple. The student can install the latest version 3.9.6 from the given link.

1. Open <https://www.python.org/downloads/> to download.
2. Click on the “[macOS link](#)” below “**Download Python 3.9.6**”





3. Choose the first link in the case of the intel processor or the second one.

Files			
Version	Operating System	Description	MD5
<a href="#">Gzipped source tarball</a>	Source release		79
<a href="#">XZ compressed source tarball</a>	Source release		ed
<a href="#">macOS 64-bit intel installer</a>	Mac OS X	for macOS 10.9 and later	d7
<a href="#">macOS 64-bit universal2 installer</a>	Mac OS X	for macOS 10.9 and later, including macOS 11 Big Sur on Apple Silicon (experimental)	93
<a href="#">Windows embeddable package (32-bit)</a>	Windows		5b
<a href="#">Windows embeddable package (64-bit)</a>	Windows		89
<a href="#">Windows help file</a>	Windows		91
<a href="#">Windows installer (32-bit)</a>	Windows		90
<a href="#">Windows installer (64-bit)</a>	Windows	Recommended	ac

**Note:** Rest of the steps of installations are similar to that of Windows OS. Follow the steps given for Python Installation on Windows.

Now that we have installed Python let us see how to run it locally.

There are multiple ways to run a python program locally like:

1. **Python Shell** (also called **repl**: read, evaluate, print and loop).
2. **Code Editors** like Visual Studio Code Editor.

Let's start by using the Python Shell.

For this we can use the system command line terminals.

To open the terminal in:

#### **Windows:**

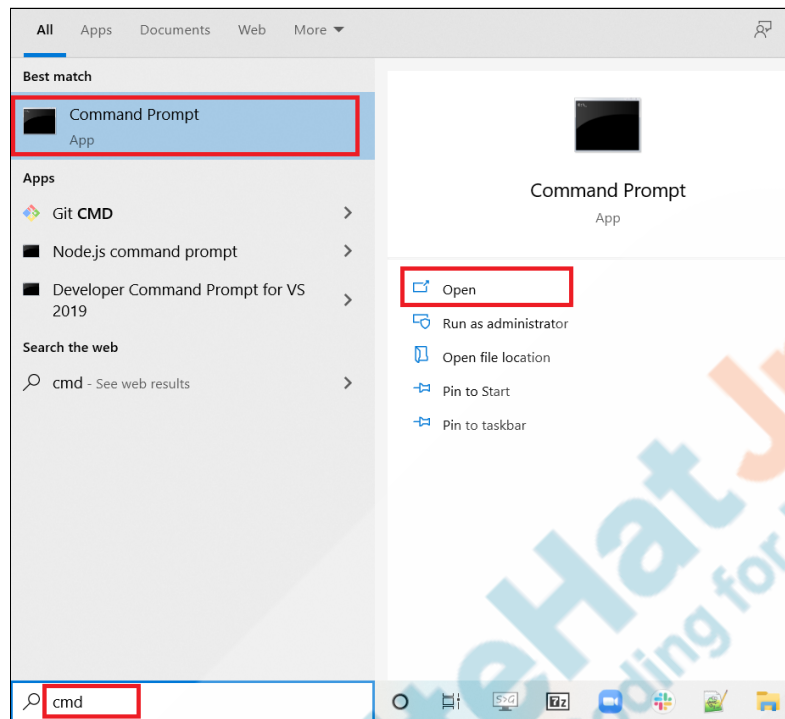
Type **cmd** in the search bar to open **Command Prompt**.

*Note: The terminal in Windows is called Command Prompt.*

#### **Mac:**

Type **terminal** in the search bar to open **Terminal**.

*The teacher opens the terminal.*



We can use the **Python3** command to enter the **Python Shell Environment**.

*The teacher types **Python3** to enter into the python shell.*

**Note:** **Python3** works if the checkbox with “**Add into system environment**” was checked, in case it was missed, then in this case run with **PY** command.

```
Command Prompt - Python3
Microsoft Windows [Version 10.0.19042.1110]
(c) Microsoft Corporation. All rights reserved.

C:\Users\preet>Python3
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

<p>As you can see, we are getting the version of Python.</p> <p>Symbol, &gt;&gt;&gt;, indicates that the Python Shell is ready to accept the command.</p> <p><u><b>Python Shell</b> only takes a single expression, evaluates it, prints it as a result to the user, and then gets ready for another input.</u></p> <p>Let me write a simple command to ask the name and greet the user.</p> <p>So which function can I use to ask for user input?</p> <p>Let's take a <b>name</b> variable to store the value of the input given by the user.</p> <p><i>The teacher types the code and presses "Enter"</i></p> <pre>&gt;&gt;&gt; name = input("Hi, What is your name ? ")</pre>	<p><b>ESR:</b> we can use <b>input()</b></p>
<pre>&gt;&gt;&gt; name = input("Hi, What is you name ? ")</pre>	
<p>The command is executed when we press <b>Enter</b>. Since we used the <b>input()</b> method to take value from the user, the Python Shell will wait until we provide the input. Let's provide the name of the person which will be stored in the <b>name</b> variable and press Enter.</p>	
<pre>&gt;&gt;&gt; name = input("Hi, What is you name ? ") Hi, What is you name ? Daisy &gt;&gt;&gt; _</pre>	

<p>We can check the value of the <b>name</b> variable using <b>print()</b> method.</p>	
<pre data-bbox="565 420 998 577">&gt;&gt;&gt; print(name) Daisy &gt;&gt;&gt;</pre>	
<p>In order to exit from the Python Shell Environment, we can use the <b>quit()</b> function.</p>	
<pre data-bbox="479 735 1096 997">&gt;&gt;&gt; name = input("Hi, What is you name ? ") Hi, What is you name ? Daisy &gt;&gt;&gt; print(name) Daisy &gt;&gt;&gt; quit() C:\Users\preet&gt;</pre>	
<p>Before we experiment more on the Python Shell, do remember in JavaScript and React Native, where we used to add various libraries in our programs?</p> <p>Can you tell me why we use libraries?</p> <p>Great!</p> <p>Python also comes with many useful predefined methods that we can use to add in our program so we do not have to write those ourselves.</p> <p>To understand how Python manages these functions/methods we need to understand what is:</p>	<p><b>ESR:</b> Yes.</p> <p><b>ESR:</b> Libraries are helper programs that have functions and properties that we can directly use in our program.</p>

- Library
- Package
- Module

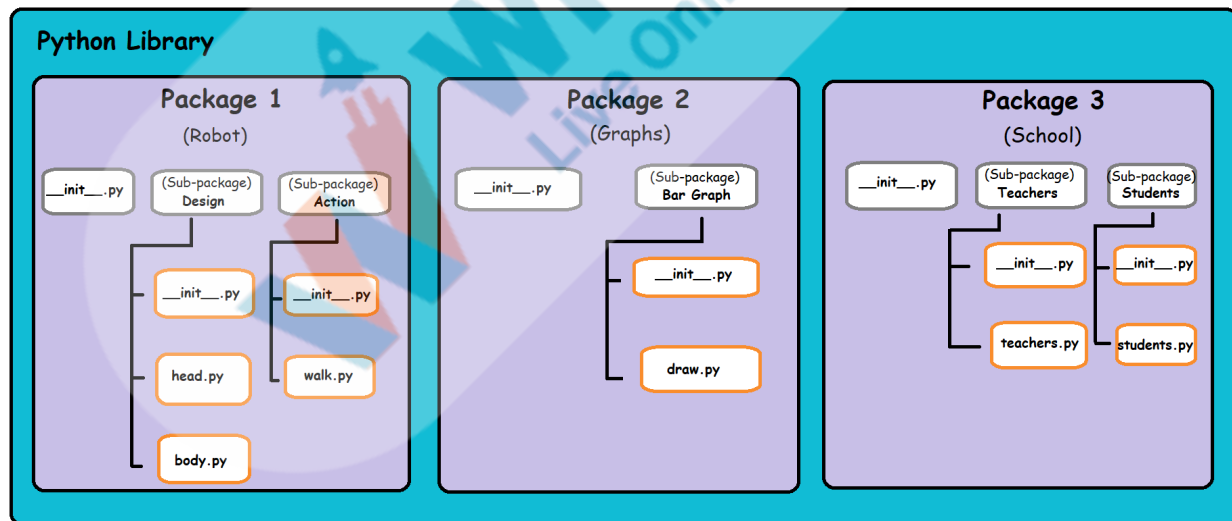
Open [Teacher Activity 3](#) to show the image to the students.

**Library:** The **library** has a collection of packages that allows users to perform many tasks without writing the code on their own.

For example, a library can have a **Robot**, **Graphs** and **School** Package.

Each package has sub-packages inside them.

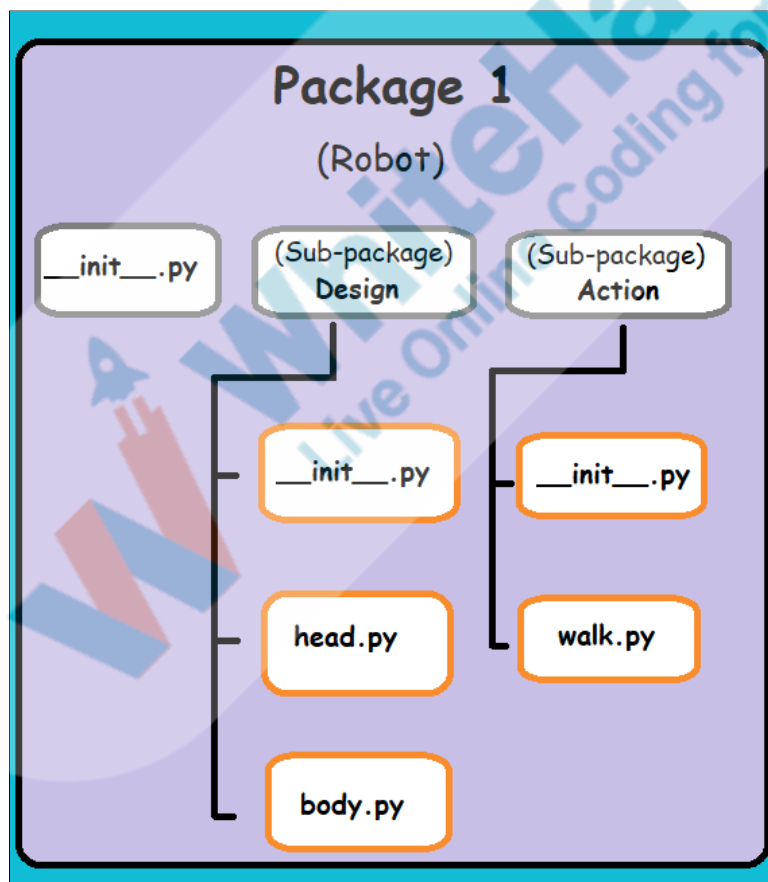
**Note:** The package/module names are only for explanation purposes. These are not part of the actual Python Library.



**Package:** The **package** is a collection of modules. A package can have one or more modules. A package also has a `__init__.py` file, which is used to distinguish a package from a module.

For example, a **Robot** is a Package.  
This package has **Design** and **Action** as the sub-packages inside them.

***Note:** The package/module names are only for explanation purposes. These are not part of the actual Python Library.*

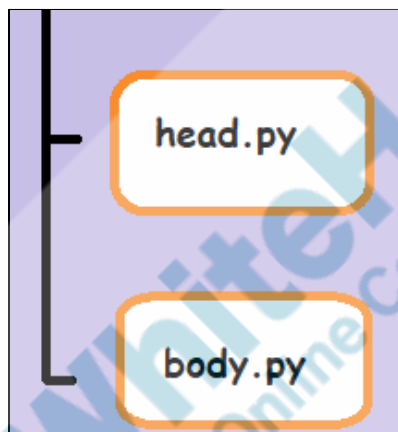


**Module:** A module is a Python file having a **.py** extension. It contains the Python statements and methods definitions that can be used in the program.

For example, a **Robot** is a Package.

This package has **Design** sub-packages which contain a **head** and **body** module.

***Note:** The package/module names are only for explanation purposes. These are not part of the actual Python Library.*



To use functions and definitions from these modules, we can add import them to our program.

Python library as many modules defined. These are called **built-in modules**. We can also write our own modules too. These are called **custom modules**.

We will learn about built-in modules today.

### **Built-in Modules:**

Do you recall us using simple functions like **print()** and **input()**?



These are built-in functions, built-in modules contain the most commonly used functions of Python.

We can import them to use in our program, instead of copying their definitions or rewriting them.

Examples of the most commonly used modules are **random**, **time**, **math**, **os**, and many more.

Built-in modules are installed along with python, which we can directly use by importing them. We will see in a while how to use them.

However, all the modules cannot be a part of Python software and need to be installed as and when required in the code.

Python has a package manager to keep all the modules together.

To manage and install all such packages and modules, Python comes with a package manager called **pip** which stands for "**Pip Installs Packages**".

**Note :** From Python 3.4 onwards we can install pip while installing the Python. Remember we checked the box with pip while installing Python.

**ESR:** Yes

To check if **pip** is correctly installed on your system:

Run **pip --version** or **pip3 --version** on your command prompt/ terminal.

```
C:\Users\preet>pip --version
pip 21.2.3 from C:\Users\preet\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\pip (python 3.9)
```

```
C:\Users\preet>
```

```
C:\Users\ADMIN\Downloads>pip3 --version
pip 21.2.3 from C:\Users\ADMIN\AppData\Local\Programs\Python\Python39\lib\site-packages\pip (python 3.9)
C:\Users\ADMIN\Downloads>
```

Now, let us check a few built-in modules in Python like **time**, **random**.

To use the methods of the **time** module, we first need to import the **time** module. We can do this using the **import** statement.

**Note:** Make sure to enter Python Shell using the “**Python3**” command as discussed before.

```
>>> import time_
```

Let's check what all methods are present in the **time** module. We can use the **dir()** method to list all the methods present in the **time** module.

```
print(dir(time))
```

**Note:** Make sure to enter Python Shell using the “**Python3**” command as discussed before.

```
>>> print(dir(time))
['_STRUCT_TM_ITEMS', '__doc__', '__loader__', '__name__', '__package__', '__spec__', 'altzone', 'asctime', 'ctime', 'daylight', 'get_clock_info', 'gmtime', 'localtime', 'mktime', 'monotonic', 'monotonic_ns', 'perf_counter', 'perf_counter_ns', 'process_time', 'process_time_ns', 'sleep', 'strptime', 'struct_time', 'thread_time', 'thread_time_ns', 'time', 'time_ns', 'timezone', 'tzname']
>>>
```

Can you locate “**localtime**”?

ESR: Yes.

The **dir()** allows us to check available definitions like “**localtime**”, “**timezone**” and so on from the **time** module.

Shall we check the **random** module?

ESR: Yes

**Note:** Make sure to enter Python Shell using the “**Python3**” command as discussed before.

```
>>> import random
>>> print(dir(random))
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST', 'SystemRandom', 'TWOPI', '_Sequence', '_Set', '_all_', '_builtins_', '_cached_', '_doc_', '_file_', '_loader_', '_name_', '_package_', '_spec_', '_acumulate', '_acos', '_bisect', '_ceil', '_cos', '_e', '_exp', '_floor', '_inst', '_log', '_os', '_pi', '_random', '_repeat', '_sha512', '_sin', '_sqrt', '_test', '_test_generator', '_urandom', '_warn', '_betavariate', '_choice', '_choices', '_exponential', '_gamma', '_gamma', '_gauss', '_getrandbits', '_getstate', '_lognorm', '_lognorm', '_normal', '_normal', '_pareto', '_pareto', '_randbytes', '_randint', '_random', '_randrange', '_sample', '_seed', '_setstate', '_shuffle', '_triangular', '_uniform', '_vonmises', '_weibull', '_weibull']
```

We can see different functions from the **random** module.

Now that we have installed Python on our system, are you excited to use these modules and build something?

ESR: Yes



I would like you to use the **time** module to create a countdown timer today.

You can share your screen.

**Teacher Stops Screen Share**

So now it's your turn.

Please share your screen with me.

<div>  </div> <p><b>Teacher Starts Slideshow</b></p> <p><b>Slide 15 to 19</b></p> <p>Refer to speaker notes and follow the instructions on each slide.</p>	
<p>We have one more class challenge for you. Can you solve it?</p> <p>Let's try. I will guide you through it.</p>	
<div>  </div> <p><b>Teacher Ends Slideshow</b></p>	
<p><b>STUDENT-LED ACTIVITY - 20 mins</b></p>	
<ul style="list-style-type: none"> <li>• Ask the student to press the ESC key to come back to the panel.</li> <li>• Guide the student to start Screen Share.</li> <li>• The teacher gets into Fullscreen.</li> </ul>	
<p><b>ACTIVITY</b></p> <ul style="list-style-type: none"> <li>• Explore howdoi</li> <li>• Create a stopwatch using the time module.</li> </ul>	
<b>Teacher Action</b>	<b>Student Action</b>
<p><i>The teacher asks the student to click on <a href="#">Student Activity 1</a> to download python for Windows/Mac based on the OS in the student's system.</i></p> <p><i>Follow the steps given in the teacher activities to guide the student. In case a student has installed Python(with <b>pip</b>) proceed with <b>howdoi</b>.</i></p>	<p><i>The student opens <a href="#">Student Activity 1</a> to install Python</i></p>
<p>Before we start, I have a question for you.</p>	

<p>You have been coding for so long now, whenever you have a question about syntax or program, what do you do?</p> <p>Yes, most of us check Google when we have any query, and in return, we get multiple links. We need to then filter or glance to get what we need, isn't it?</p> <p>Python has a handy package, for this reason, called <b>howdoi</b>.</p> <p>As the name suggests, we can ask any syntax-related question to this package.</p> <p>Let us install that package for now.</p> <p>Type <b>pip3 install howdoi</b>.</p>	<p><b>ESR:</b> Varied / Google</p> <p><b>ESR:</b> Yes</p> <p><i>The student installs the howdoi package.</i></p>
 <pre> C:\Users\ADMIN\AppData\Local\Programs\Python\pip3 install howdoi Requirement already satisfied: howdoi in c:\users\admin\appdata\local\programs\python\pyt ) Requirement already satisfied: Pygments in c:\users\admin\appdata\local\programs\python\p howdoi) (2.9.0) Requirement already satisfied: cssselect in c:\users\admin\appdata\local\programs\python\ n howdoi) (1.1.0) Requirement already satisfied: lxml in c:\users\admin\appdata\local\programs\python\pytho doi) (4.6.3) Requirement already satisfied: pyquery in c:\users\admin\appdata\local\programs\python\py howdoi) (1.4.3) Requirement already satisfied: requests in c:\users\admin\appdata\local\programs\python\p howdoi) (2.25.1) </pre>	
<p>The message displayed will tell us if <b>howdoi</b> was installed properly.</p> <p><b>howdoi</b> is used for searching functionalities related to Python programming language.</p>	

The result is not always perfect. But it is a helpful and quick tool to learn to do anything related to Python programming.

Let's ask some python related questions to **howdoi** :

### Syntax:

```
>>> howdoi <write a command in English language to search>
```

*The student explains the response of howdoi to the teacher.*

*The text in this < > is of the user's choice.*

Example:

**howdoi write python function**

*The teacher asks the student to explain the output.*

```
C:\Users\ADMIN\AppData\Local\Programs\Python>howdoi write python function
def foo(bar):
    """
    Takes bar and does some things to it.
    """
    return bar
```

Wow, see we got to see the entire syntax to write a definition along with how to use it.

In the example;

- A function is defined with the name as **foo()** with as the **bar** parameter.
- Followed by a : (**colon**)
- The steps given inside the function are indented.

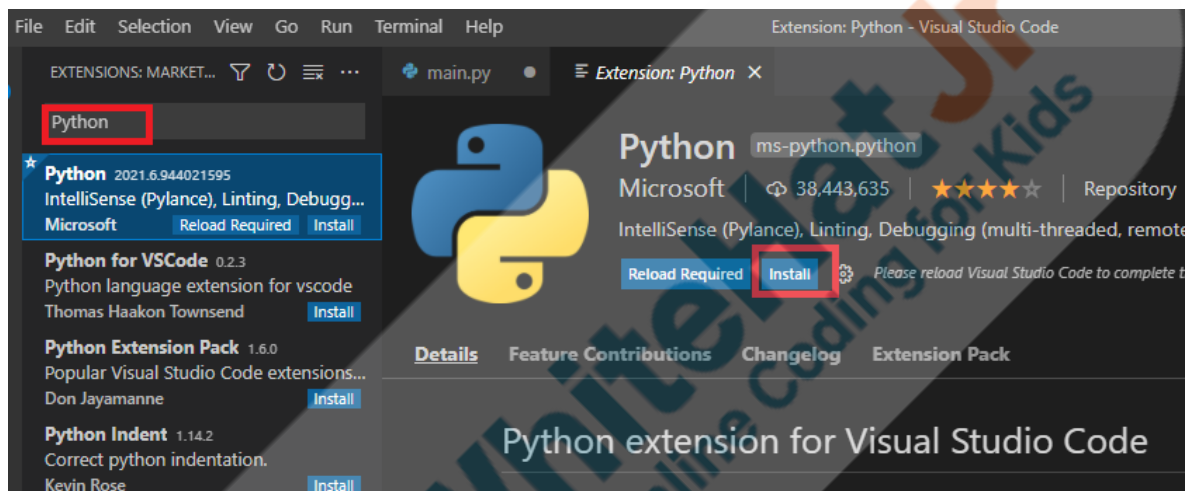
You can experiment with it more after the class.

Now we will set up the Visual Studio Code Editor to run the python code.

Open Visual Studio Code Editor(VSC). I will guide you to install Python Extension in the VSC.

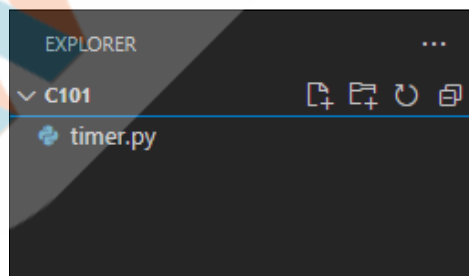
*The teacher can refer to [Teacher Activity 2](#) to guide the student for installing Python Extension in the VSC.*

*The student installs the Python extension in VSC.*



1. Create a folder named **C101** in your local system.
2. Open that folder in VSC.
3. Create a new file named **timer.py**. Remember, a python file will always have a **.py** extension.

*The student creates a new file, renames it **timer.py**.*



Have you ever used a Timer?

**ESR: Varied**

<p><i>If time permits, the teacher can ask the student to check out the timer on the mobile.</i></p> <p>What happens in the timer?</p> <p>Perfect!</p> <p>So which module can we use for this project?</p> <p>To start with, first, we will import the <b>time</b> module.</p>	<p><b>ESR:</b> We set the timer, once we start the timer it will start the countdown until it reaches 00:00. Then the alarm rings.</p> <p><b>ESR:</b> We can use the <b>time</b> module.</p>
<pre># import the time module import time</pre>	
<p>First set the number of seconds for which we want to run the timer, correct?</p> <p>So, which built-in function shall we use to ask time from the user?</p> <p>Let's create a variable to store the number of seconds entered by the user.</p>	<p><b>ESR:</b> Yes</p> <p><b>ESR:</b> We will use <b>input()</b></p>
<pre># input time in seconds seconds = input("Enter the time in number of seconds: ")</pre>	
<p>Great, what should we do next?</p> <p>We will display the time entered by the user in <b>min:sec</b>(minutes colon seconds) format, and then we will gradually reduce the time counter till it reaches 0.</p> <p>We will create a separate function for the countdown, let us name it <b>countdown_timer()</b>.</p>	<p><b>ESR:</b> Varied</p>



<p>Do you remember the syntax for defining a function?</p> <p><i>The teacher can guide the student to create a function, check for the syntax, and guide him/her where needed.</i></p>	<p><b>ESR:</b> Varied.</p> <p><i>The student creates a function <code>countdown_timer()</code>.</i></p>
<pre># define the countdown timer function def countdown_timer(seconds):</pre>	
<p>We have received the input from the user in seconds; let us first convert it into minutes and seconds:</p> <ol style="list-style-type: none"> <li>1. Create a variable <b>min</b>; divide input seconds by <b>60</b> and assign to <b>min</b>.</li> <li>2. Create a variable <b>sec</b>; divide input seconds by <b>60</b> to get a reminder and save it in <b>sec</b>. Use % modulo operator for the remainder.</li> </ol> <p>Now we can combine <b>min</b> and <b>sec</b> to print the timer in <b>min: sec</b>(e.g. 05:46) format using <b>f-strings</b>:</p> <p><code>f '{min}:{sec}'</code></p> <p><b>f-strings</b> are string literals that have an <b>f</b> at the beginning and curly braces containing variable names that will be replaced with their actual values.</p> <ol style="list-style-type: none"> <li>3. Assign combined string value to a variable <b>timer</b></li> <li>4. Print the <b>timer</b>.</li> </ol>	

```
def countdown_timer(seconds):
```

```
    mins = int(seconds / 60)
    secs = int(seconds % 60)

    timer = f'{mins}:{secs}'

    print(timer)
```

Let's call this function and pass seconds to it.

The **input()** function always returns a **str** type, so make sure to convert it into **int** while passing to the function.

```
# input time in seconds
seconds = input("Enter the time in number of seconds: ")

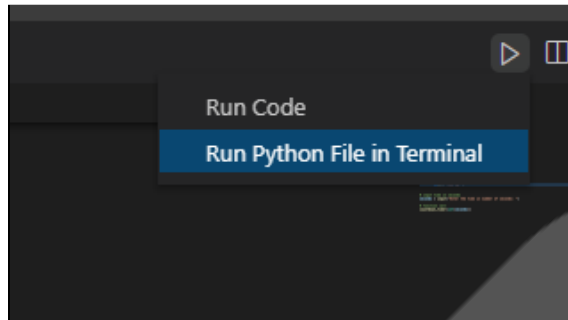
# function call
countdown_timer(int(seconds))
```

Shall we run the code to check the output?

**ESR:** Yes

There are 3 ways to run the Python code:

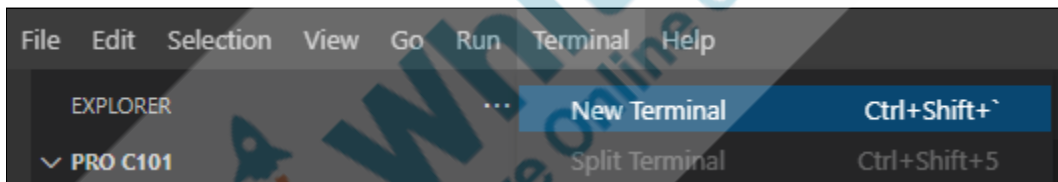
1. **Option 1:** Using **VSC Run Button** in the top left corner. This will open the VSC Terminal automatically to run the Python code at the bottom of the VSC.



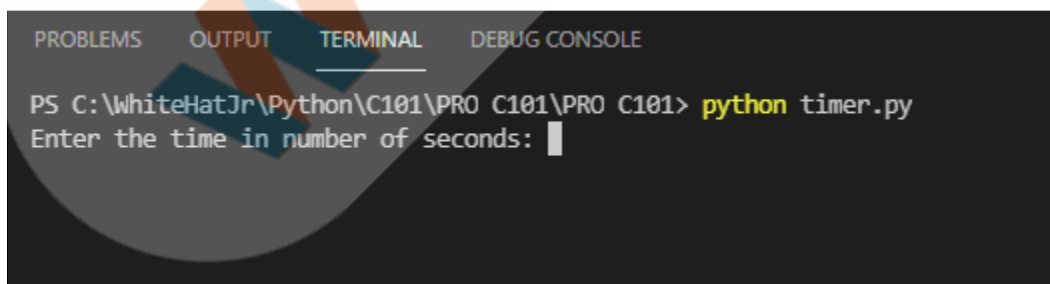
2. **Option 2:** Using the **python <filename.py>** command in VSC Terminal:

- Click on the “**Terminal**” and then “**New Terminal**” in the Menu Bar at the top.
- Run the python **<filename.py>** command in the terminal.

Click on the “**Terminal**” and then “**New Terminal**” in the Menu Bar at the top.

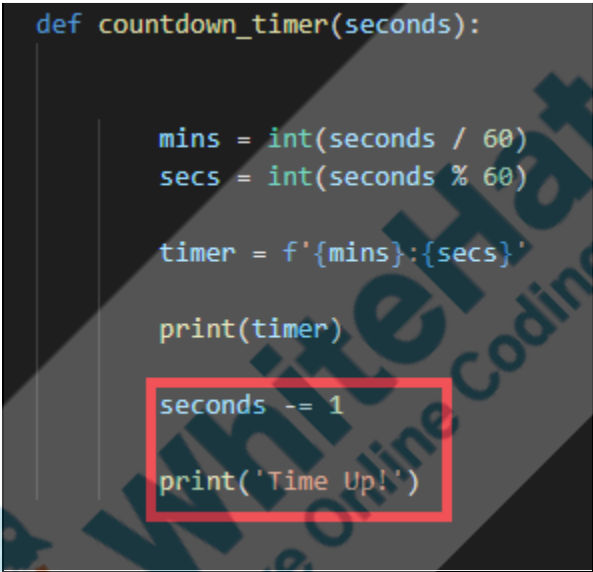
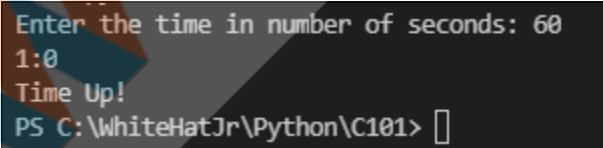


Run the python **<filename.py>** command in the terminal



3. **Option 3:** Using cmd(Windows) & system terminal(Mac):

<ul style="list-style-type: none"> <li>○ <b>Windows:</b> <ul style="list-style-type: none"> <li>■ Open cmd window, navigate to the folder which contains the .py file.</li> <li>■ Run the <b>python &lt;filename.py&gt;</b> command in the cmd window.</li> </ul> </li> <li>○ <b>Mac:</b> <ul style="list-style-type: none"> <li>■ Open Terminal.</li> <li>■ Run the <b>python &lt;filename.py&gt;</b> command in the Terminal.</li> </ul> </li> </ul>	
<p><b>Windows:</b></p> <p><i>To run the code cmd in windows, navigate to the folder which contains the .py file. Run the <b>python &lt;filename.py&gt;</b> command in the cmd window.</i></p> <pre>C:\WhiteHatJr\Python\C101&gt;python timer.py Enter the time in number of seconds: _</pre> <p><b>Mac:</b></p> <pre>((base) mac@MACs-MBP Downloads % python timer.py Enter the time in number of seconds: _</pre>	
<p><b>Output :</b></p> <p>Let us enter 60 as an input and check the output:</p> <pre>Enter the time in number of seconds: 60 1:0 PS C:\WhiteHatJr\Python\C101&gt;  </pre>	
<p>What do you see?</p>	<p><b>ESR:</b> The time gets displayed into min:sec</p>

<p>Yes, what should we do next?</p> <p>Yes, we will reduce the time and when the time is 0 we will print(<b>"Time Up!"</b>).</p> <p>So why don't you code next to reduce the time?</p>	<p>format.</p> <p><b>ESR:</b> Reduce the time by seconds.</p>
	
<p>Shall we run the code again to check?</p>	<p><b>ESR:</b> Yes.</p>
	
<p>The timer did not start. Can you tell me what the issue is?</p> <p>Yes, if you notice we are reducing the time by 1 second just once, and then we are printing <b>"Time Up!"</b>. So it does not serve our purpose. We need to keep reducing time till it becomes 0, and keep printing the new</p>	<p><b>ESR:</b> The time did not reduce.</p>

<p>value.</p> <p>How can we achieve repetitive work?</p> <p>Yes, but instead of a <b>for</b> loop today, we will use a <b>while</b> loop.</p> <p>A <b>while</b> loop statement in Python repeatedly executes a code block as long as a given condition is true.</p> <p>Do you have a wall clock which runs on battery?</p> <p>The clock will keep ticking till their batteries are live, i.e till power in batteries are “True” as soon as batteries die, the status will be “False” and the clock will stop.</p> <p>Let’s understand how to use while loops in Python.</p> <p><b>Syntax:</b></p> <div><pre>while expression:     statement(s)</pre></div>	<p><b>ESR:</b> Using <b>for</b> loops.</p> <p><b>ESR:</b> Varied</p>
--	--

**Syntax:**

```
while condition:  
    statement(s)
```

The while loop syntax has:

- The **while** keyword
- A condition (that is, an expression that evaluates to True or False)
- A colon(:)
- An indented block of code

*The teacher can explain while loop with [Teacher Activity 4](#).*

*Open [Teacher Activity 4](#) and run the code cell.*

```
# Print the number till it is > 0  
  
num = 5  
  
while num > 0:  
    print(num)  
    num = num -1  
  
5  
4  
3  
2  
1
```

We want to keep reducing time till the seconds value is greater than 0. Hence, we can use a **while** loop here.

Let me help you with the syntax.

*Guide the student to write the code.*

*Please make sure the code is indented properly, post adding the while loop.*

```
# define the countdown timer function
def countdown_timer(seconds):
    while seconds > 0:
        mins = int(seconds / 60)
        secs = int(seconds % 60)

        timer = f'{mins}:{secs}'
        print(timer)

        seconds -= 1

    print('Time Up!')
```

As you see now, the seconds will keep reducing till it becomes zero.

We aligned “**Time Up!**” with a while loop; hence, it will be executed only once the loop is over.

Let us run the code one more time.

```
0:10
0:9
0:8
0:7
0:6
0:5
0:4
0:3
0:2
0:1
Time Up!
PS C:\WhiteHatJr\Python\C101> |
```



It was very fast, and the output was very long as it is printing every time the seconds get reduced.

What should happen here?

Ideally, the time should be displayed at the same place, and it should be overwritten after every second, right?

So, let us make a few final changes to our code.

1. The time module has a function called **sleep()**. This halts the execution of the code for the time specified. We will use **sleep()** to hold for **1 sec** before the next time is printed. Because **sleep()** is part of the time. We use it as **time.sleep(1)**.
2. In order to print in the same place instead of the next line; we can add **end=\r** in **print(timer)**. **\r** will help replace the current string with a new string post the **end** of first string

**ESR:** Varied

**ESR:** Yes

```
def countdown_timer(seconds):
    while seconds:
        mins = int(seconds / 60)
        secs = int(seconds % 60)

        timer = f'{mins}:{secs}'

        print(timer, end='\r')
        time.sleep(1)
        seconds -= 1

    print('Time Up!')
```

Let us run the code one more time to see the changes.

**Note :** The expected output can be seen at [Link](#).

*The student runs the code.*

```
PS C:\WhiteHatJr\Python\C101> py timer.py
Enter the time in number of seconds: 5
Time Up!
```

Amazing! Isn't it?

Do you see with importing a built-in module of time; it was so easy to create a timer?

You did amazing work today! We installed Python, PIP package manager, **howdoi** and also created a program using the built-in **time** module.

**ESR:** Yes.

**ESR:** Varied

**Teacher Guides Student to Stop Screen Share**

**WRAP UP SESSION - 5 mins**

**Teacher Starts Slideshow**  
Slide 20 to 25



### Activity details

**Following are the WRAP-UP session deliverables:**

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

**WRAP-UP QUIZ**  
Click on In-Class Quiz

**Continue WRAP-UP Session**  
Slide 26 to 31



### Activity Details

**Following are the session deliverables:**

© 2021 - WhiteHat Education Technology Private Limited.

Note: This document is the original copyright of WhiteHat Education Technology Private Limited.

Please don't share, download or copy this file without permission.

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

### FEEDBACK

- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

#### Teacher Action

You get hats-off for your excellent work!

In the next class, we learn about the Python os and shutil module to manage files in the operating system.

#### Student Action

*Make sure you have given at least 2 Hats Off during the class for:*

Creatively Solved Activities  +10

Great Question  +10

Strong Concentration  +10

### PROJECT OVERVIEW DISCUSSION

Refer the document below in Activity Links Sections

Teacher Clicks

✕ End Class

### ADDITIONAL ACTIVITIES

Now, since we have some additional time, let's perform an additional activity wherein we play a sound when the timer stops.

Let us add a sound when we display the message “Time Up!”.

*If additional activity is performed, share the GitHub link from [Teacher Activity 6](#) with the student over chat.*

1. Download the sound file from the shared link.
2. Save it in the same folder 101.

There is a module to play sound in Python. It is known as the **playsound** module.

Let us check how to import it using **howdoi**.

Type “**howdoi import playsound module**”

```
C:\Users\ADMIN\AppData\Local\Programs\Python\Python39>howdoi import playsound module
from playsound import playsound
playsound('myfile.wav')
```

Import **playsound** in **timer.py**.



```
timer.py > ...
1 # import the time module
2 import time
3 from playsound import playsound
4
```

Do you see the wavy line below the **playsound**?

**ESR:** Yes

That is because **playsound** is not an inbuilt module in Python, we will have to install it to use it.

Let us run **pip install playsound** in the terminal.

```
> pip install playsound
```

Once it is successfully installed, we will use the <b>playsound()</b> function to run the sound.	
<pre>print('Time Up!') playsound('mixkit-sound.wav')</pre>	
Let us run the code once again.	
The sound plays once the message “ <b>Time Up!</b> ” is printed	
We shall stop here, you can experiment more with <b>howdoi</b> , see you in the next class!	

ACTIVITY LINKS		
Activity Name	Description	Link
Teacher Activity 1	Link to Download Python	<a href="#">Download Python</a>
Teacher Activity 2	Reference for Setting Up VSC to Run Python Code	<a href="https://docs.google.com/document/d/1t-iy_Yaj5Ptc5tMLt54hufnPgGWFGa5BIKcNlt-euE/edit">https://docs.google.com/document/d/1t-iy_Yaj5Ptc5tMLt54hufnPgGWFGa5BIKcNlt-euE/edit</a>
Teacher Activity 3	Understanding Python Library, Packages & Modules	<a href="https://s3-whjr-curriculum-upload.s.whjr.online/e3ced16b-23a7-4285-a390-d834766d6ee5.png">https://s3-whjr-curriculum-upload.s.whjr.online/e3ced16b-23a7-4285-a390-d834766d6ee5.png</a>
Teacher Activity 4	While Loop Examples	<a href="https://colab.research.google.com/drive/1RMVui0fsxoqwVio7Z_yvR40V-FhbO_-y?usp=sharing">https://colab.research.google.com/drive/1RMVui0fsxoqwVio7Z_yvR40V-FhbO_-y?usp=sharing</a>
Teacher Activity 5	Teacher Reference Code	<a href="https://github.com/procodingclass/PRO-C101-Teacher-Reference-Code">https://github.com/procodingclass/PRO-C101-Teacher-Reference-Code</a>
Teacher Activity 6	Sound File	<a href="https://github.com/pro-whitehatjr/C101_StudentActivity1_Soundfile">https://github.com/pro-whitehatjr/C101_StudentActivity1_Soundfile</a>
Teacher Activity 7	Teacher Reference Code With	<a href="https://github.com/procodingclass">https://github.com/procodingclass</a>

	Additional Activity	<a href="#">/PRO-C101-Teacher-Reference-Code-AA</a>
Student Activity 1	Link to Download Python	<a href="#">Download Python</a>
Teacher Reference 1	Project Document	<a href="https://s3-whjr-curriculum-upload.s.whjr.online/13666fbf-8010-4b7f-a304-26b62ba2fc4f.pdf">https://s3-whjr-curriculum-upload.s.whjr.online/13666fbf-8010-4b7f-a304-26b62ba2fc4f.pdf</a>
Teacher Reference 2	Project Solution	<a href="https://github.com/procodingclass/PRO-C101-Project-Solution/tree/main">https://github.com/procodingclass/PRO-C101-Project-Solution/tree/main</a>
Teacher Reference 3	Visual-Aid	<a href="https://s3-whjr-curriculum-upload.s.whjr.online/af9fe33a-790a-4762-9f6c-7bdef6248b3e.html">https://s3-whjr-curriculum-upload.s.whjr.online/af9fe33a-790a-4762-9f6c-7bdef6248b3e.html</a>
Teacher Reference 4	In-Class Quiz	<a href="https://s3-whjr-curriculum-upload.s.whjr.online/7f14c00c-b6c7-4dd9-9e9e-defe4067ca2a.pdf">https://s3-whjr-curriculum-upload.s.whjr.online/7f14c00c-b6c7-4dd9-9e9e-defe4067ca2a.pdf</a>