Topic	**AUTOMATE FILE SEGREGATION**	
Class Description	**The student will be introduced to os and shutil modules of Python. The student builds a python program to move image files from one folder to another folder.**	
Class	**PRO C102**	
Class time	**45 mins**	
Goal	● Understanding the os and shutil modules ● Use the **os** and **shutil** module to create a folder organizer program.	
Resources Required	● Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Visual Studio Code ● Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Visual Studio Code	
Class structure	**Warm-Up** **Teacher-led Activity 1** **Student-led Activity 1** **Wrap-Up**	**05 mins** **15 mins** **20 mins** **05 mins**

WARM-UP SESSION - 05 mins

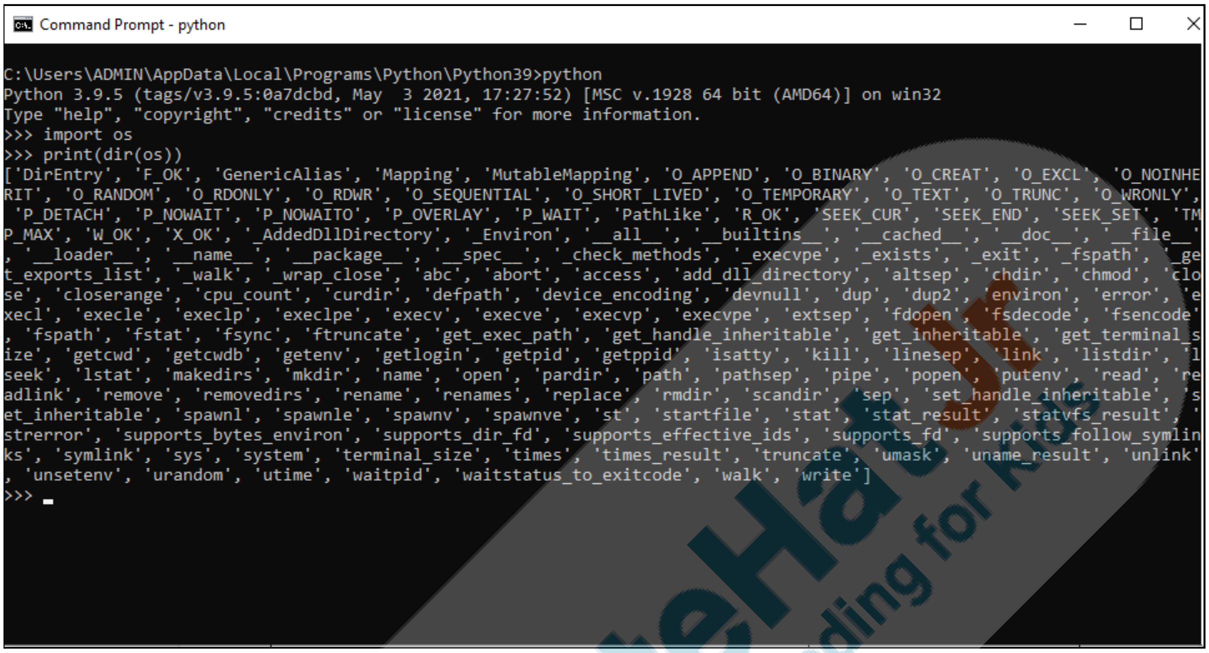
Teacher Starts Slideshow
Slide 1 to 3
Refer to speaker notes and follow the instructions on each slide.

Hey <student's name>. How are you? It’s great to see you! Are you excited to learn something new today?	**ESR**: Hi, thanks! Yes I am excited about it!

Following are the WARM-UP session deliverables: ● Greet the student. ● Revision of previous class activities. ● Quizzes.	Click on the slide show tab and present the slides
WARM-UP QUIZ Click on In-Class Quiz	
Continue WARM-UP Session **Slide 4 to 15**	
Following are the session deliverables: ● Appreciate the student. ● Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.	
Teacher Ends Slideshow	
TEACHER-LED ACTIVITY - 15 mins	
Teacher Initiates Screen Share	
<u>ACTIVITY</u> ● **Introduce the os and shutil module of Python.** ● **Explore various functionality of os and shutil modules.**	
Teacher Action	**Student Action**
Do you know when you boot up your laptop/computer, it is actually running a program? Do you know what the program is called?	**ESR**: Varied **ESR:** Operating System Windows macOS Linux

Yes! The operating system. Our operating system provides us with different functionalities like moving files, copying files, and so on. The **os** module in Python provides us a way of using these operating system-dependent functionality. Are you excited to try out? Let’s get started.	**ESR:** Yes
In Python, we use **import** to use a module/library in our project. *The teacher can refer to Teacher Activity 1 for the **os** module methods.* The **os** is a built-in module hence need not require to install it, we can directly import it. Let us use the Python Shell to experiment with a few functions of **os** and **shutil** module *The teacher opens the shell terminal and enters the Python Shell Environment by typing **python** or **py**.* Now, let us import **os** `>>> import os` We can use **dir()** on the **os** module to see the methods of the **os** module. To print the result we can use:	.

```
>>> print(dir(os))
```



```
C:\Users\ADMIN\AppData\Local\Programs\Python\Python39>python
Python 3.9.5 (tags/v3.9.5:0a7dcbd, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> print(dir(os))
['DirEntry', 'F_OK', 'GenericAlias', 'Mapping', 'MutableMapping', 'O_APPEND', 'O_BINARY', 'O_CREAT', 'O_EXCL', 'O_NOINHE
RIT', 'O_RANDOM', 'O_RDONLY', 'O_RDWR', 'O_SEQUENTIAL', 'O_SHORT_LIVED', 'O_TEMPORARY', 'O_TEXT', 'O_TRUNC', 'O_WRONLY',
 'P_DETACH', 'P_NOWAIT', 'P_NOWAITO', 'P_OVERLAY', 'P_WAIT', 'PathLike', 'R_OK', 'SEEK_CUR', 'SEEK_END', 'SEEK_SET', 'TM
P_MAX', 'W_OK', 'X_OK', '_AddedDllDirectory', '_Environ', '__all__', '__builtins__', '__cached__', '__doc__', '__file__'
, '__loader__', '__name__', '__package__', '__spec__', '_check_methods', '_execvpe', '_exists', '_exit', '_fspath', '_ge
t_exports_list', '_walk', '_wrap_close', 'abc', 'abort', 'access', 'add_dll_directory', 'altsep', 'chdir', 'chmod', 'clo
se', 'closerange', 'cpu_count', 'curdir', 'defpath', 'device_encoding', 'devnull', 'dup', 'dup2', 'environ', 'error', 'e
xecl', 'execle', 'execlp', 'execlpe', 'execv', 'execve', 'execvp', 'execvpe', 'extsep', 'fdopen', 'fsdecode', 'fsencode'
, 'fspath', 'fstat', 'fsync', 'ftruncate', 'get_exec_path', 'get_handle_inheritable', 'get_inheritable', 'get_terminal_s
ize', 'getcwd', 'getcwdb', 'getenv', 'getlogin', 'getpid', 'getppid', 'isatty', 'kill', 'linesep', 'link', 'listdir', 'l
seek', 'lstat', 'makedirs', 'mkdir', 'name', 'open', 'pardir', 'path', 'pathsep', 'pipe', 'popen', 'putenv', 'read', 're
adlink', 'remove', 'removedirs', 'rename', 'renames', 'replace', 'rmdir', 'scandir', 'sep', 'set_handle_inheritable', 's
et_inheritable', 'spawnl', 'spawnle', 'spawnv', 'spawnve', 'st', 'startfile', 'stat', 'stat_result', 'statvfs_result', '
strerror', 'supports_bytes_environ', 'supports_dir_fd', 'supports_effective_ids', 'supports_fd', 'supports_follow_symlin
ks', 'symlink', 'sys', 'system', 'terminal_size', 'times', 'times_result', 'truncate', 'umask', 'uname_result', 'unlink'
, 'unsetenv', 'urandom', 'utime', 'waitpid', 'waitstatus_to_exitcode', 'walk', 'write']
>>> 
```

As you can see there are many functionalities in the **os** module like **open, rename, replace, mkdir,** and many more.

Let us try a few of them.

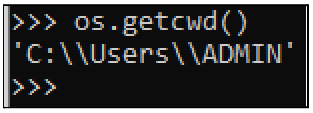
First let's check the folder/directory in which we are working.

We can also check our current working directory by using **getcwd()** in the **os** module.

Current Working Directory is the directory/ folder in which we are currently working to make changes(like create/updating/copying/renaming/removing a file).

The teachers run the command in the Python Shell.

```
>>> os.getcwd()
```



```
>>> os.getcwd()
'C:\\Users\\ADMIN'
>>>
```

Now let’s create one new folder(directory) in this our current working directory.

We can use the **mkdir()** method in the **os** module to create new files or folders.

Syntax:

```
os.mkdir("name of the file/folder to create")
```

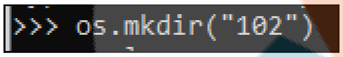
Example:

The teachers run the command in the Python Shell.

```
>>> os.mkdir("102")
```

This will create a folder named **C102** in the current working directory.

Note*: Remember to use “ “ for the folder name as it is a string value.*



```
>>> os.mkdir("102")
```

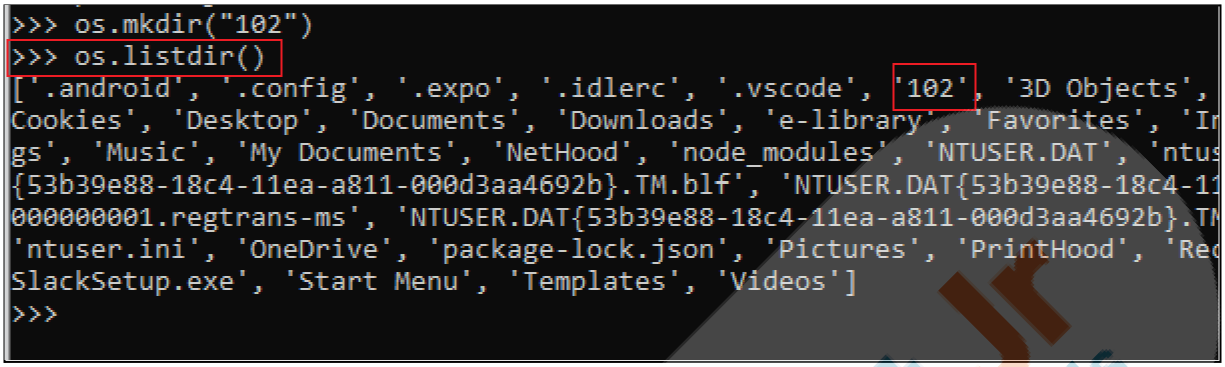
How do we check if this folder is created?

To get all the files and folders of the current directory, we can use the **listdir()** method in the **os** module.

The teachers run the command in the Python Shell.

ESR: Varied

```
>>> os.listdir()
```



```
>>> os.mkdir("102")
>>> os.listdir()
['.android', '.config', '.expo', '.idlerc', '.vscode', '102', '3D Objects',
Cookies', 'Desktop', 'Documents', 'Downloads', 'e-library', 'Favorites', 'In
gs', 'Music', 'My Documents', 'NetHood', 'node_modules', 'NTUSER.DAT', 'ntus
{53b39e88-18c4-11ea-a811-000d3aa4692b}.TM.blf', 'NTUSER.DAT{53b39e88-18c4-11
000000001.regtrans-ms', 'NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TM
'ntuser.ini', 'OneDrive', 'package-lock.json', 'Pictures', 'PrintHood', 'Rec
SlackSetup.exe', 'Start Menu', 'Templates', 'Videos']
>>>
```

To check if a particular file or folder is present in a given folder path, we can use the **exists()** method in the **os.path** module.

Syntax:

```
os.path.exists(“path of a folder/file)
```

Example**:**

The teachers run the command in the Python Shell.

1. Take a **path** variable and assign a folder/file path different from the files/folders present in the current working directory.

```
>>> path = '/usr/local/bin/'
```

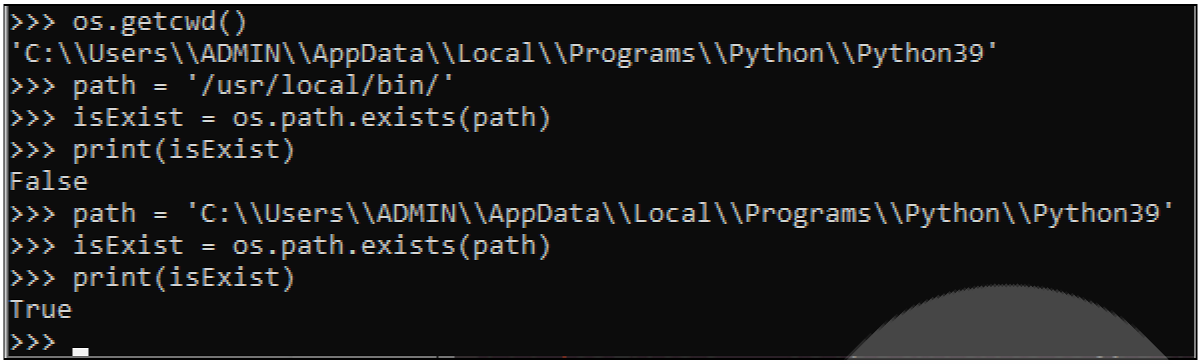
2. Take **isExist** variable use **exists()** method in **os.path** module.

```
>>> isExist = os.path.exists(path)
```

3. Print the **isExist** variable.

```
>>> print(isExist)
```

*The terminal prints **False**; Try the same commands by copying the path with the result of **getcwd()** and check the result as **True**.*



```
>>> os.getcwd()
'C:\\Users\\ADMIN\\AppData\\Local\\Programs\\Python\\Python39'
>>> path = '/usr/local/bin/'
>>> isExist = os.path.exists(path)
>>> print(isExist)
False
>>> path = 'C:\\Users\\ADMIN\\AppData\\Local\\Programs\\Python\\Python39'
>>> isExist = os.path.exists(path)
>>> print(isExist)
True
>>> 
```

Note: The teacher can download images from Teacher Activity 2 and save them inside the Downloads folder.

We learn to use a few more methods in **os** and **shutil.** Let us try to understand the **splittext()** method in the **os.path** module.

Syntax:

```
root, ext = os.path.splitext(“path of a folder/file)
```

The **os.path.splitext()** method in Python is used to split the path name into the (root, ext) pair where root is the whole path except the extension and ext is the file extension with a dot.

Example:

The teachers run the command in the Python Shell.

1. Import os module

```
>>> import os
```

2. Take a **path** variable and assign a path of a file present in the systems “Downloads” folder.

Please check the path based on where you are currently in the terminal.

```
>>> path = 'Downloads\feather.jfif'
```

3. Take two variables **root** and **extension** and split the path in root and extension of the file/folder path using **splittext()** method in the **os.path** module.

   Here is a Python trick, instead of writing two lines for creating two variables, one for the file name and one for the extension, we can create both in just one line separated by a comma (,)

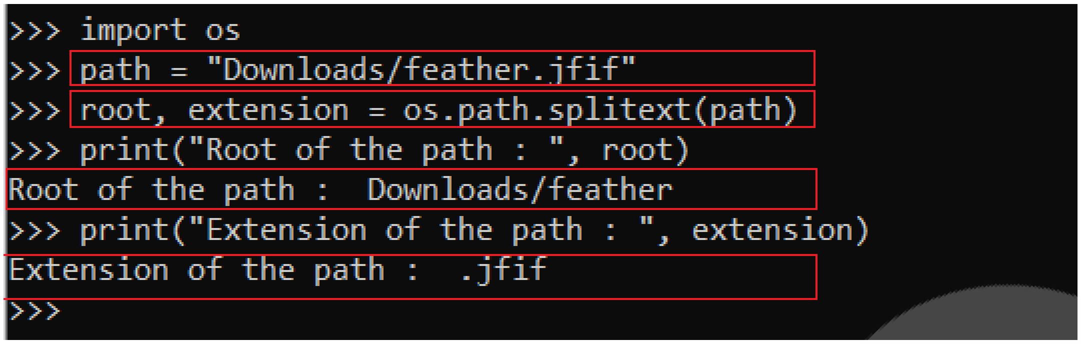
```
>>> root, extension = os.path.splitext(path)
```

4. Print **root** and **extension** of the specified path.

```
>>> print("Root of the path : ", root)

>>> print("Extension of the path : ",
extension)
```

***Note 1**: If the specified path has a leading period ('.'), it will be ignored.*

***Note 2**: If the file name is given without extension, the second **print()** will throw an error.*



```
>>> import os
>>> path = "Downloads/feather.jfif"
>>> root, extension = os.path.splitext(path)
>>> print("Root of the path : ", root)
Root of the path :  Downloads/feather
>>> print("Extension of the path : ", extension)
Extension of the path :  .jfif
>>>
```

Now we are going to see how we can make a copy of a particular file.

We can perform many basic operations on individual files and folders using the **os module**.

There is another module named **shutil,** The **shutil** module offers a number of high-level operations on files and collections of files.

This module can help us in **automating** the process of copying and removal of files and directories.

We will make use of a few functions from **os** and a few from **shutil** module.

Let’s check out the files in the “Downloads” folder first:

Example:-

The teachers run the command in the Python Shell.

1. Import **os** and **shutil** module.

```
>>> import os
>>> import shutil
```

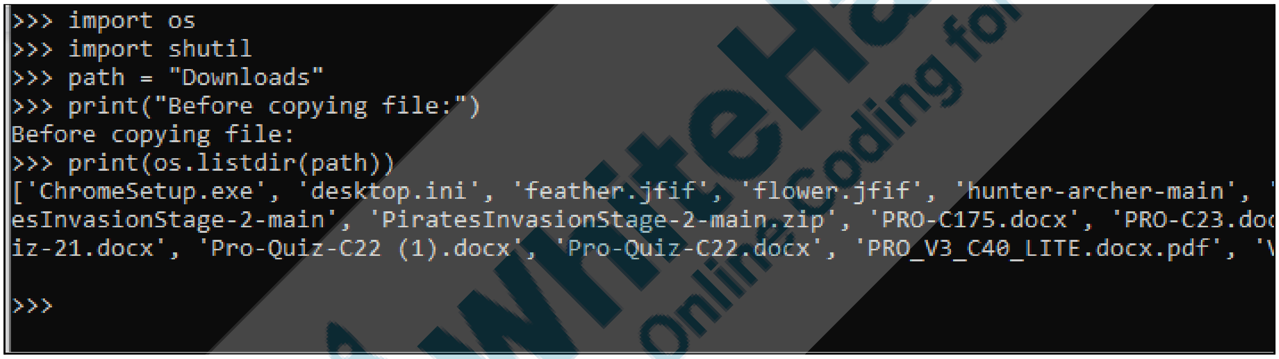
2. Take a **path** variable and assign a path to the “**Downloads**” folder in your system.

```
>>> path = 'Downloads'
```

3. List files and directories using **print()** method.

```
>>> print("Before copying file:")

>> print(os.listdir(path))
```

*This allows us to view the content from the “**Downloads**” folder in your system*



```
>>> import os
>>> import shutil
>>> path = "Downloads"
>>> print("Before copying file:")
Before copying file:
>>> print(os.listdir(path))
['ChromeSetup.exe', 'desktop.ini', 'feather.jfif', 'flower.jfif', 'hunter-archer-main', '
esInvasionStage-2-main', 'PiratesInvasionStage-2-main.zip', 'PRO-C175.docx', 'PRO-C23.doc
iz-21.docx', 'Pro-Quiz-C22 (1).docx', 'Pro-Quiz-C22.docx', 'PRO_V3_C40_LITE.docx.pdf', 'V

>>>
```

Let us pick up one of the files from this set of data, and we will create a copy of it:

Example:-

The teachers run the command in the Python Shell.

1. Take a **source** variable and assign a file path.

```
>>> source = "/Downloads/feather.jfif"
```

2. Take a **destination** variable, assign a path where we want to create a copy.

```
>>> destination = "/Downloads/copyfeature.jfif"
```

3. Use the **copy()** method in the shutil **module** to copy the file from source to destination.

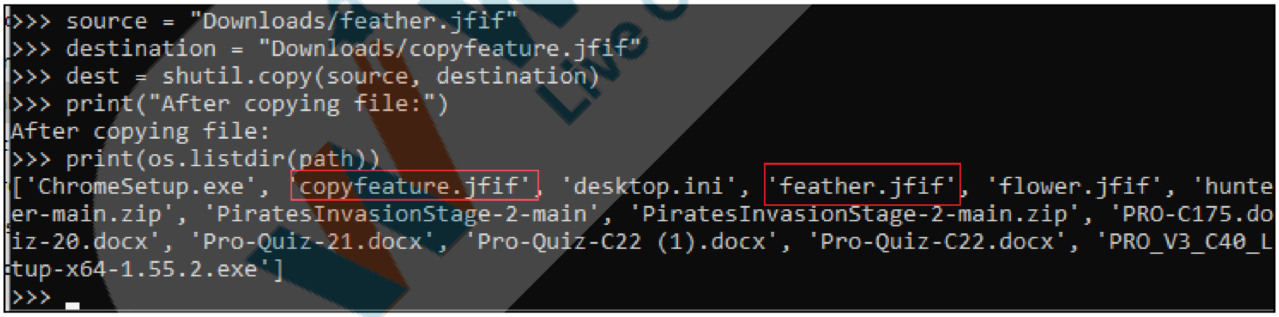
   Syntax:

```
shutil.copy(source, destination)
```

```
>>> dest =shutil.copy(source, destination)
```

4. Print the content of the folder to see if the copy is created.

```
>>> print("After copying file:")

>>> print(os.listdir(path))
```



```
>>> source = "Downloads/feather.jfif"
>>> destination = "Downloads/copyfeature.jfif"
>>> dest = shutil.copy(source, destination)
>>> print("After copying file:")
After copying file:
>>> print(os.listdir(path))
['ChromeSetup.exe', 'copyfeature.jfif', 'desktop.ini', 'feather.jfif', 'flower.jfif', 'hunte
er-main.zip', 'PiratesInvasionStage-2-main', 'PiratesInvasionStage-2-main.zip', 'PRO-C175.do
iz-20.docx', 'Pro-Quiz-21.docx', 'Pro-Quiz-C22 (1).docx', 'Pro-Quiz-C22.docx', 'PRO_V3_C40_L
tup-x64-1.55.2.exe']
>>> 
```

The teacher can also show a copy of the file in the folder as well.

There are other methods of **shutil()** to move the files or remove files altogether.

There is another method named **shutil.move()** to move a file from source to destination. We will use many such functionalities in upcoming classes and understand the details while using them. For now, where do you think we can make use of these modules, to make our task easy? We keep downloading so much information almost daily, and all those files by default get saved in a **Downloads** folder. How about we create code, on running, which can, 1. Identify file type. 2. Move them in different folders based on their type. Wouldn’t it make our folders more organized? Python is used widely to create such automation projects due to its wide libraries. Why don’t you write a code using the **os** and **shutil** module to move the image files from the download folder to a different folder?	**ESR:** Varied **ESR:** Yes **ESR:** Yes
Teacher Stops Screen Share	
So now it’s your turn. Please share your screen with me.	
Teacher Starts Slideshow **Slide 16 to 20** Refer to speaker notes and follow the instructions on each slide.	
We have one more class challenge for you. Can you solve it?	

Let's try. I will guide you through it.	

Teacher Ends Slideshow

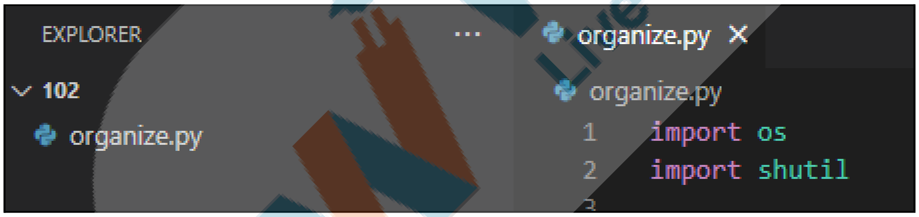
STUDENT-LED ACTIVITY - 20 mins

- **Ask the student to press the ESC key to come back to the panel.**
- **Guide the student to start Screen Share.**
- **The teacher gets into Fullscreen.**

ACTIVITY

- **Create code to move image files from the Downloads folder to the new folder.**

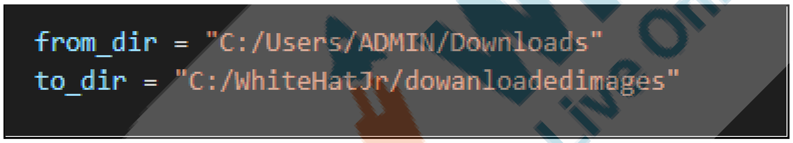
Teacher Action	Student Action
You can start with creating a folder named **102**, and open it in Visual Studio Code. Create a new file named **organize.py** Start with importing **os** and **shutil** module.	*The student creates a file named **organise.py** and imports **os** and **shutil** module.*



```
import os
import shutil
```

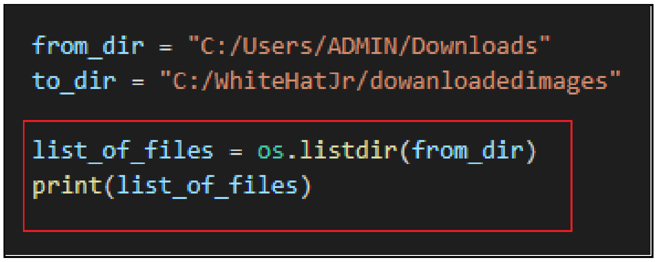
Great! Before we write further code, what do you think we should do to move all image files from the Downloads folder to another folder? Correct!	**ESR:** Give source and destination path.

Which function can be used to create a folder? Yes, you can use **os.mkdir()** to create a new folder. However, if we add it to the code, it will create a new folder every time we will run this code. Do we want that? So let us create a folder wherever we want to save our image files. *The teacher can help the student to create a new folder in his system. Make sure to keep that folder out of the* ***Downloads*** *folder.* Let's store these two folder path into variables: 1. Take two variables, **from_dir** and **to_dir** to store source path and destination path, respectively. ***Note****: In terminal path is given with " \ " but VSC accepts path as " / ".* *Modify the path name as per your system.*	**ESR:** we can use **os.mkdir().** **ESR:** No



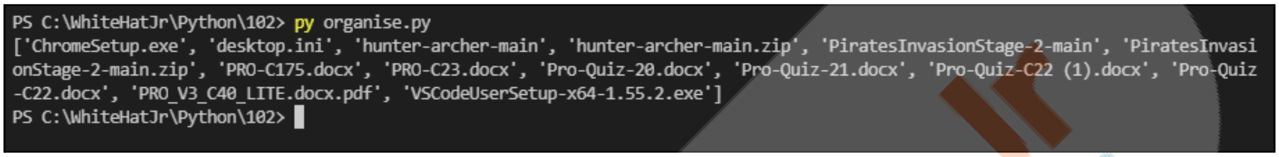
```
from_dir = "C:/Users/ADMIN/Downloads"
to_dir = "C:/WhiteHatJr/dowanloadedimages"
```

Now we need to get names of all the files from the folder of the given path, so we can use **os.listdir()**: 2. Take a variable, **list_of_files**, to store the names of all the files. Let us see the content, which functions shall we use? 3. Add **print(list_of_files)** and run the code.	**ESR:** we can use **print(list_of_files).**



```
from_dir = "C:/Users/ADMIN/Downloads"
to_dir = "C:/WhiteHatJr/dowanloadedimages"

list_of_files = os.listdir(from_dir)
print(list_of_files)
```

OUTPUT:



```
PS C:\WhiteHatJr\Python\102> py organise.py
['ChromeSetup.exe', 'desktop.ini', 'hunter-archer-main', 'hunter-archer-main.zip', 'PiratesInvasionStage-2-main', 'PiratesInvasi
onStage-2-main.zip', 'PRO-C175.docx', 'PRO-C23.docx', 'Pro-Quiz-20.docx', 'Pro-Quiz-21.docx', 'Pro-Quiz-C22 (1).docx', 'Pro-Quiz
-C22.docx', 'PRO_V3_C40_LITE.docx.pdf', 'VSCodeUserSetup-x64-1.55.2.exe']
PS C:\WhiteHatJr\Python\102> █
```

As you can see, the content of the folder is saved as a **list** type.

We can also notice that the download folder has many kinds of files, text files, pdfs, images, executable files, etc.

How do we traverse through the **list**?

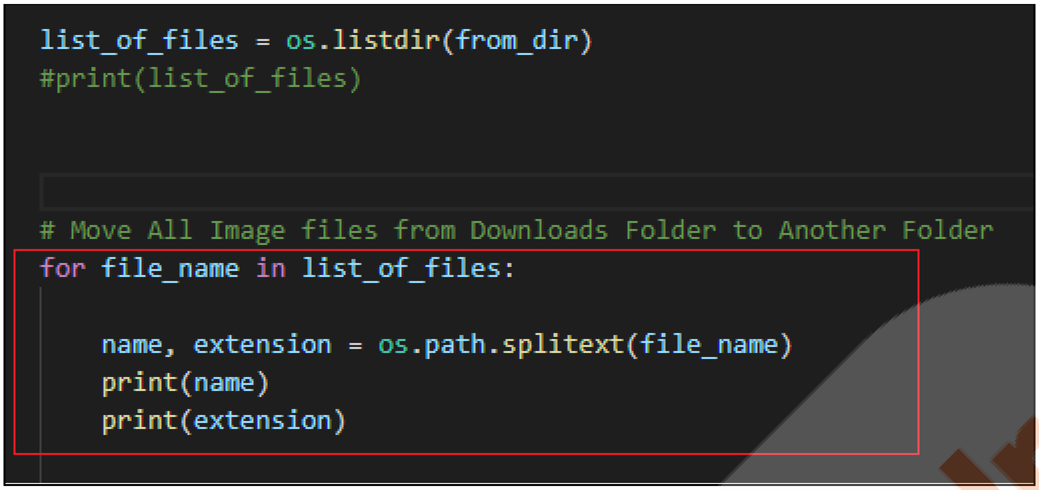
Great!

4. Create a **for-in** loop to traverse through the **list_of_files**:

   Use **os.path.splitext()** on each file name to find the extension.

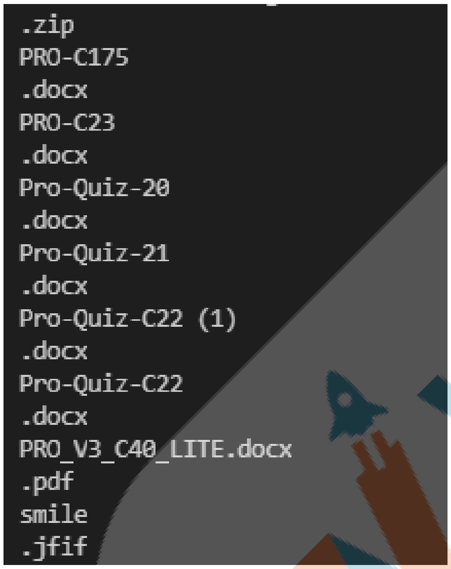
   You can comment by adding # to the previous **print()** statement, and add a new one after **splittext().**

ESR: We can use a **for-in loop**.



```
list_of_files = os.listdir(from_dir)
#print(list_of_files)


# Move All Image files from Downloads Folder to Another Folder
for file_name in list_of_files:

    name, extension = os.path.splitext(file_name)
    print(name)
    print(extension)
```

OUTPUT:



```
.zip
PRO-C175
.docx
PRO-C23
.docx
Pro-Quiz-20
.docx
Pro-Quiz-21
.docx
Pro-Quiz-C22 (1)
.docx
Pro-Quiz-C22
.docx
PRO_V3_C40_LITE.docx
.pdf
smile
.jfif
```

We can see each file name and extension are printed one below the other.

What will happen if there is a folder inside Downloads, which does not have an extension?

The extension will remain blank.

How do we recognize if the file is an image file?

Yes, the image file can have any of the following extensions: ['.gif', '.png', '.jpg', '.jpeg','.jfif']

For today, we will focus only on the image files.

For each file, we will check if the extension is blank or if it has an extension that matches the set of image extensions.

Let us write the conditions for this:

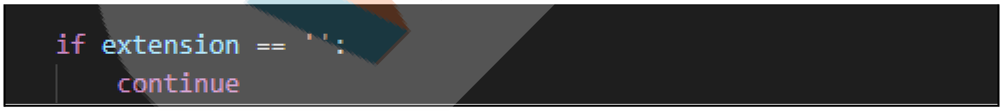
1. Write an **if** condition to check if the extension is blank, if the condition is **true** then **continue**.

   The **"continue"** statements are used inside loops. When the program execution reaches a **"continue"** statement, the program execution immediately jumps back to the start of the loop and reevaluates the loop's condition.

   if the **extension** is **blank**; it will jump to the next file and check for the extension of the next file.

ESR: Varied

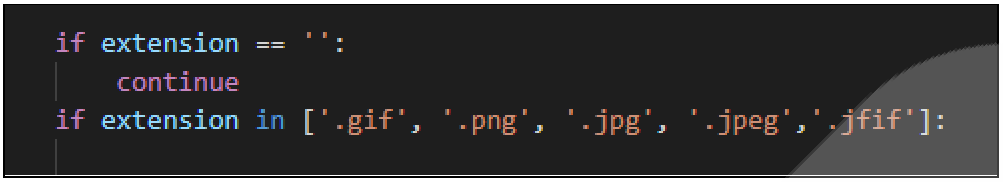
ESR: Image file has the **.png** or **.jpg** extension



```
if extension == '':
    continue
```

Now we will write:

2. Another **if** condition to check if the **extension** of the image file in the path is one of the extensions in a **list [‘.gif’, ‘.png’, ‘.jpg’, ‘.jpeg’, ‘.jfif’]**



```
if extension == '':
    continue
if extension in ['.gif', '.png', '.jpg', '.jpeg','.jfif']:
```

If the extension is in the **list [‘.gif’, ‘.png’, ‘.jpg’, ‘.jpeg’, ‘.jfif’],** then:

3. Create 3 variables for the name of the directory paths:

    - Create **path1** as the name of the source path.

      Use string concatenation to merge **from_dir+’/’+file_name**

      Example:
      path1: Downloads/ImageName1.jpg

    - Create **path2** as we want to create a new folder with that extension name and move the files to that folder.

      Use string concatenation to merge **to_dir + '/' + "Image_Files"**

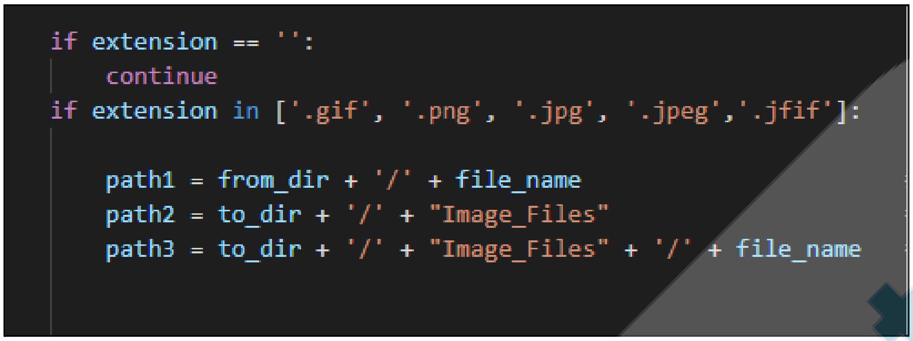
      Example:
      path2: D:/My Files/Image_Files

    - Create **path3** to assign the destination path with the same file name as the source.

      Use string concatenation to merge **to_dir + '/' + "Image_Files" + '/' +**

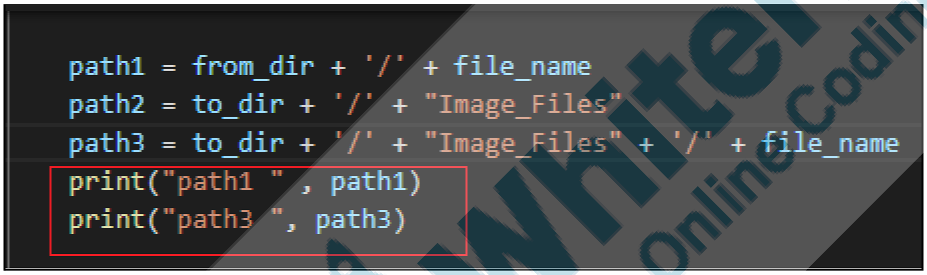
file_name

Example:
path3:
D:/MyFiles/Image_Files/ImageName1.jpg



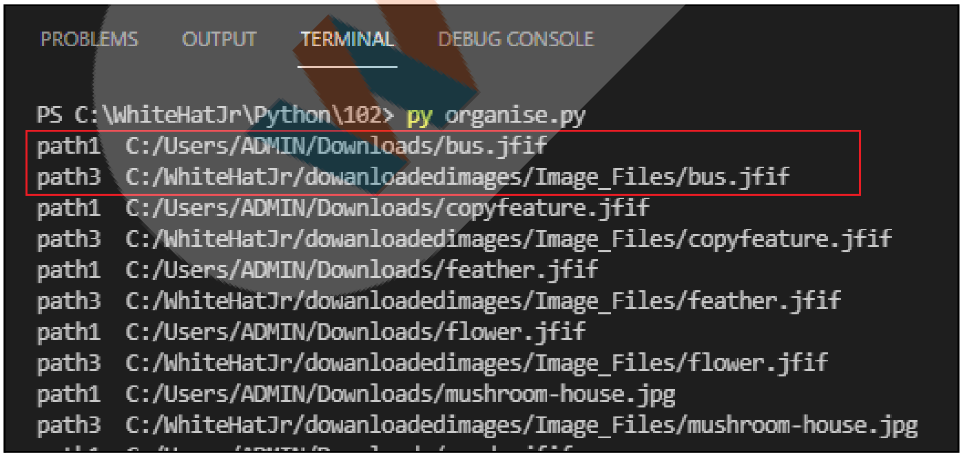
```
if extension == '':
    continue
if extension in ['.gif', '.png', '.jpg', '.jpeg','.jfif']:

    path1 = from_dir + '/' + file_name
    path2 = to_dir + '/' + "Image_Files"
    path3 = to_dir + '/' + "Image_Files" + '/' + file_name
```

Again, you can comment or remove the previous **print()** and add another to check **path1** and **path3**.



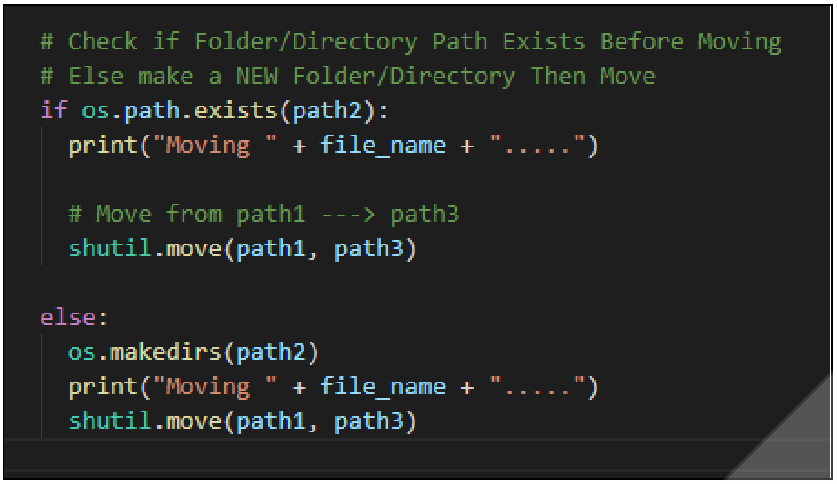
```
path1 = from_dir + '/' + file_name
path2 = to_dir + '/' + "Image_Files"
path3 = to_dir + '/' + "Image_Files" + '/' + file_name
print("path1 " , path1)
print("path3 ", path3)
```

OUTPUT:



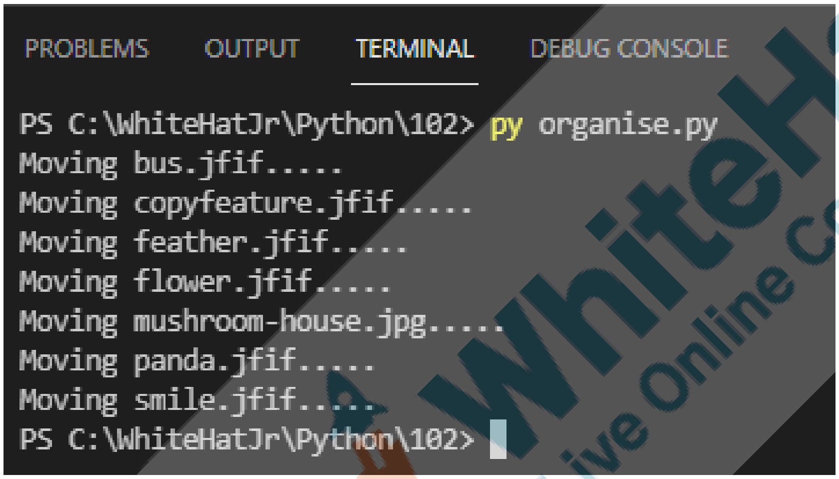
```
PS C:\WhiteHatJr\Python\102> py organise.py
path1  C:/Users/ADMIN/Downloads/bus.jfif
path3  C:/WhiteHatJr/dowanloadedimages/Image_Files/bus.jfif
path1  C:/Users/ADMIN/Downloads/copyfeature.jfif
path3  C:/WhiteHatJr/dowanloadedimages/Image_Files/copyfeature.jfif
path1  C:/Users/ADMIN/Downloads/feather.jfif
path3  C:/WhiteHatJr/dowanloadedimages/Image_Files/feather.jfif
path1  C:/Users/ADMIN/Downloads/flower.jfif
path3  C:/WhiteHatJr/dowanloadedimages/Image_Files/flower.jfif
path1  C:/Users/ADMIN/Downloads/mushroom-house.jpg
path3  C:/WhiteHatJr/dowanloadedimages/Image_Files/mushroom-house.jpg
```

We can see that the names of the **path from which the image file will be sent & the name of the path where the image file will be sent** are correct! Currently, we have used **print()** to check if the path is getting created. But what if the path of the directory/folder where we want to send the image file does not exist, then what should we do? Yes. Correct! So before moving any files, we should check if the directory/folder is present or not. 1. Check if the folder/directory path exists before moving using an **if** condition: a. Create a condition to check if the destination path exists at **path2**. b. If true, use **print()** method to print a message moving with the file name which is being moved. c. Use **shutil.move(path1, path3)**. In this case, **path1** is the source path and **path3** is the destination path. 2. Else make a new folder/directory then move: a. Use **os.makedirs()** to create **path2.** b. Use **print()** method to print a message moving with the file name. c. Use **shutil.move(path1, path3)**	**ESR:** We should use the **os.path.exists()** method to check whether the path is there or not.



```
# Check if Folder/Directory Path Exists Before Moving
# Else make a NEW Folder/Directory Then Move
if os.path.exists(path2):
  print("Moving " + file_name + ".....")

  # Move from path1 ---> path3
  shutil.move(path1, path3)

else:
  os.makedirs(path2)
  print("Moving " + file_name + ".....")
  shutil.move(path1, path3)
```

OUTPUT:



```
PS C:\WhiteHatJr\Python\102> py organise.py
Moving bus.jfif.....
Moving copyfeature.jfif.....
Moving feather.jfif.....
Moving flower.jfif.....
Moving mushroom-house.jpg.....
Moving panda.jfif.....
Moving smile.jfif.....
PS C:\WhiteHatJr\Python\102> 
```

Superb! Can you open the folders and check if files have moved? Wouldn't it help to keep our folders more organized? You can create similar code for the files with other extensions too. For now, let us revise our concepts and attempt a fun quiz.	**ESR:** Yes **ESR:** Yes

Teacher Guides Student to Stop Screen Share

WRAP UP SESSION - 5 mins

Teacher Starts Slideshow
Slide 21 to 24

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 25 to 30

Activity Details:
Following are the session deliverables:

- 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	Student Action
Excellent! You get Hats off for your excellent work! In the next class, you will be introduced to the watchdog module to manage file system creation events. You will automate "Download" folder/files movement as soon as	*Make sure you have given at least 2 Hats Off during the class for:* Creatively Solved Activities +10

the file is downloaded, and segregate it into different directories based on the type of the file.	Great Question +10 Strong Concentration +10
PROJECT OVERVIEW DISCUSSION Refer the document below in Activity Links Sections	
Teacher Clicks	✖ End Class

ACTIVITY LINKS

Activity Name	Description	Link
Teacher Activity 1	The **os** module	https://docs.python.org/3/library/os.html
Teacher Activity 2	Image Files	https://github.com/pro-whitehatjr/C102_assets
Teacher Activity 3	Teacher Reference Code	https://github.com/pro-whitehatjr/C102_TeacherReferenceCode
Student Activity 1	Images	https://github.com/pro-whitehatjr/C102_assets
Teacher Reference 1	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/34e1ea10-d6bf-4fea-bd76-1ac7878ba7a4.pdf
Teacher Reference 2	Project Solution	https://github.com/procodingclass/PRO-C102-Project-Solution
Teacher Reference 3	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/d9279460-718f-4f4a-b597-bc68211ab41e.html
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/8dd5d599-974c-4192-a548-59f7625a49b2.pdf