How to automate Outlook emails with Python

Integrate Outlook with Python applications to create, compose, and send scheduled emails. Here's how to automate Outlook emails with Python.

Python can automate many operations and you can achieve the desired functionality with just a few lines of code. This language has become popular in many fields, from data analysis to system administration.

Outlook is a popular email client. You can use Python's Outlook automation features to send emails quickly without too much effort.

The following are details on how to automatically send emails from Python using Microsoft Outlook.

Necessary conditions

To get started, download and install the following programs:

1. **Microsoft Outlook** : You can use Outlook with Gmail or any other email provider.
2. **Win32com.client** : Third-party library to connect Microsoft apps.

Install win32com.client

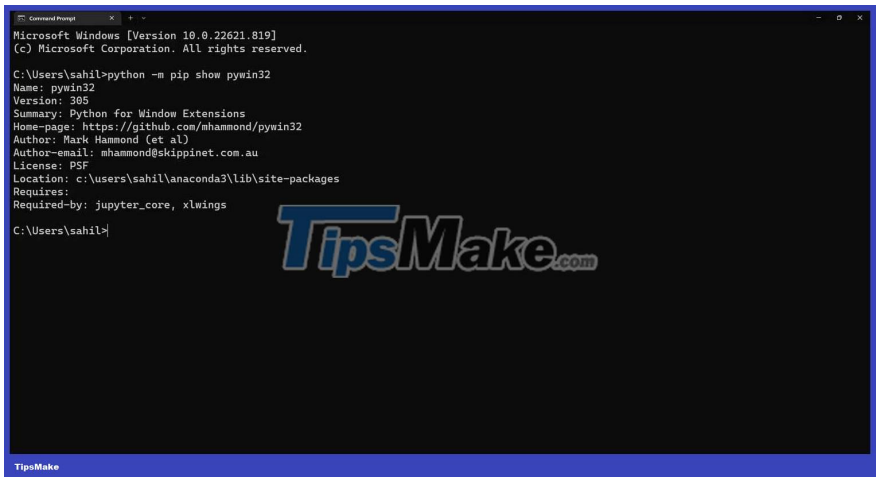
Win32com.client is an integral part of this code. You need a full nwang library to establish a connection between Microsoft Outlook and Python.

Check win32com.client version

Before installing, you can check to see if **win32com** is installed on your computer. Some IDE versions provide this package by default. You can check to see whether it is available with the following command:

```
python -m pip show pywin32
```

After running the above command, if you get a version number, you don't need to reinstall it.



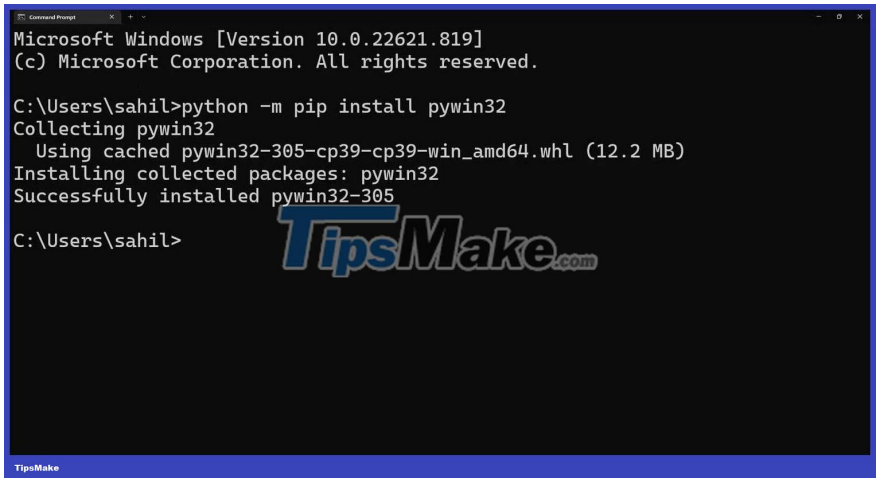
Then, if you receive the following error message, you need to install the win32com library on the system:

```
'pywin32' is not recognized as an internal or external command, operable program
```

Install win32com library

Open the prompt and enter the pip command to install the library from a terminal window.

```
python -m pip install pywin32
```



Follow the on-screen instructions to complete this process. You can use the show installation command first to verify whether win32com has been successfully installed on the system or not.

```
python -m pip show pywin32
```

Send email from Python using Outlook

Microsoft Outlook is one of the oldest and most widely used email clients. It is in the top most popular email providers today, just behind Gmail and Yahoo. The automatic link between Outlook and Python is not surprising. With just some basic tweaks, you can easily send emails quickly.

After meeting the above conditions, it's time to start writing code to automatically send emails from Outlook using Python. **To get started, you need to import the win32com.client** library using the **import** t command.

```
import win32com.client
```

Now you can write code that connects Python and the Microsoft Outlook email client.

```
ol = win32com.client.Dispatch('Outlook.Application')
```

In there:

1. **ol** contains the connection reference.
2. **Win32com.client** is a Windows library that you can use to establish a connection between Python and Outlook.
3. **Dispatch** is the connection constructor.
4. **Outlook.Application** is the name of the connection application.

Next, you need to determine the size of the new email notification so that python understands where the content needs to be updated.

```
# kích th??c c?a email m?i olmailitem = 0x0
```

In there:

Olmailitem : New variable to hold dimensions.

0x0 : The size of the new email message in Python's memory.

Python functions will open a new email item, as soon as you specify the email body size.

```
newmail = ol.CreateItem(olmailitem)
```

In there:

1. **Newmail** : New variable to hold the new email reference.
2. **ol** : Reference of a previously created connection between Python and Outlook.
3. **CreateItem(olmailitem)** : Command to create a new email draft.

Every email needs a subject line. You can define it in your code so that Python automatically adds it before sending the email to the recipient.

```
newmail.Subject = 'Testing Mail'
```

In there:

1. **Newmail** : Variable to store the new mail item reference.
2. **Subject** : Can vary, depending on what you want as the subject for the email.

You can add desired recipients in the **To** and **CC** sections as follows:

```
newmail.To = 'xyz@example.com' newmail.CC = 'xyz@example.com'
```

In there:

1. **To** : Primary recipient's email address.
2. **CC** : Email recipient was copied.

Likewise, you can even add a BCC name, in case you want to send an anonymous email to the recipient. All you have to do is move the following command behind the CC command:

```
newmail.BCC = 'xyz@example.com'
```

Python's Outlook does not limit Outlook's inherent capabilities and features. For example, even if you are using Python to manage email client responses, you can still send emails to multiple recipients. Just add a separator (;) between the email ID in the To/CC/BCC column. It's done!

Finally, after defining the subject and recipient, you can add the email content to a new mail item before sending it to the recipient in the **To** and **CC** columns .

```
= 'Hello, this is a test email to showcase how to send emails from Python and Ou
```

To add attachments to emails, you can use the following command:

```
attach = 'C:UsersadminDesktopPythonSample.xlsx' newmail.Attachments.Add(attach)
```

Now the email is ready to send. You have two options to use. If you want to preview an email before sending it to the recipient, you can use the **Display()** command as follows:

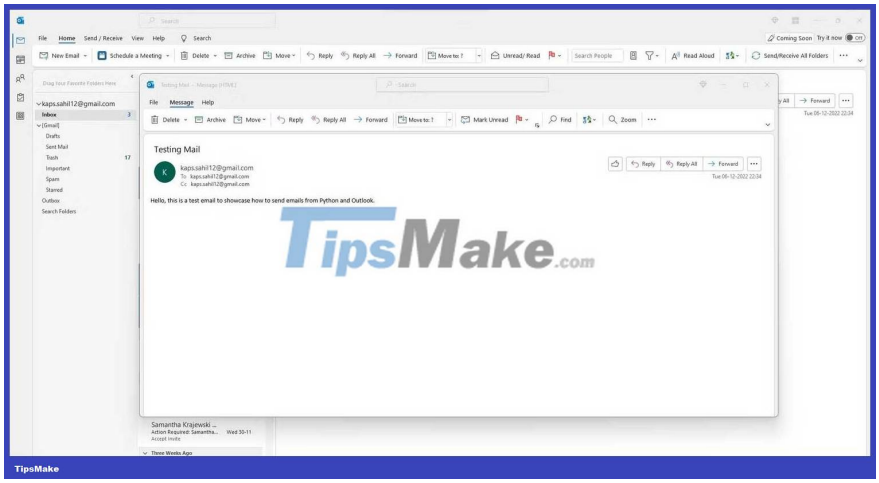
```
newmail.Display()
```

Additionally, if you don't want to preview the email and send it directly, you can replace the **Display()** command with **Send()** .

```
newmail.Send()
```

Here is the completed code:

```
import win32com.client ol=win32com.client.Dispatch("outlook.application") olmail
?n mail tr??c khi g?i nó ?i # newmail.Display() newmail.Send()
```



After successfully deploying the code, you can automatically send emails. In case an error occurs, you will see a notification at the time the email is sent. At this time, you should find out the cause to find a suitable solution.

You finished reading the article "**How to automate Outlook emails with Python**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.