 implementation.md

Prerequisite

Install python

Follow the guide to install python3 here: <https://www.python.org/downloads/>

you can verify if python is installed correctly by running the following command.

```
python --version
```

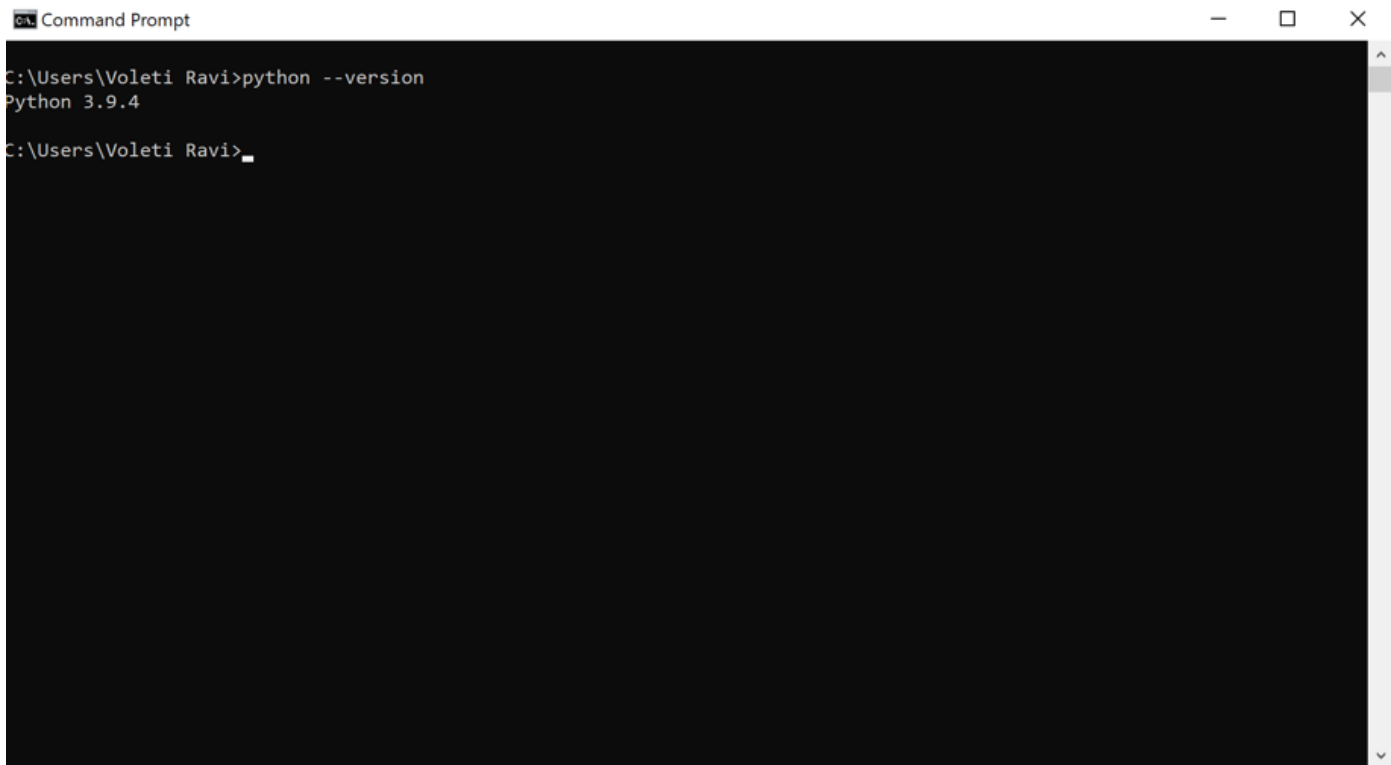
It should look like this.

Mac:



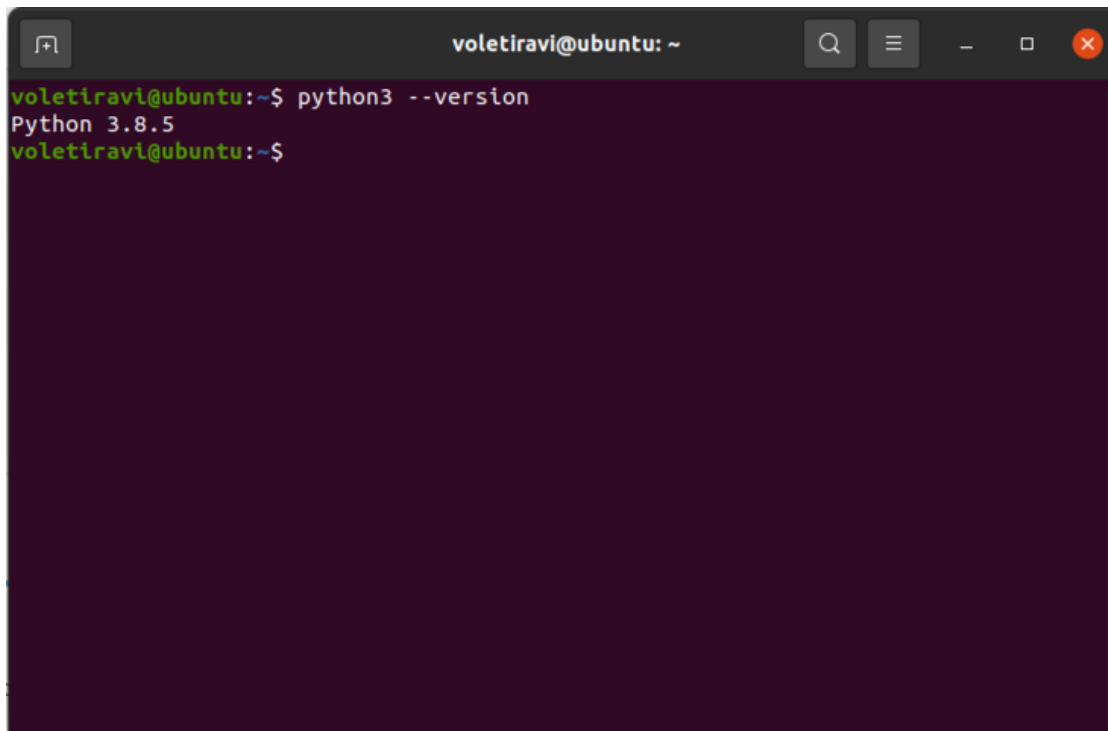
```
voletiravi@Voletis-MBP ~ % python3 --version
Python 3.9.1
voletiravi@Voletis-MBP ~ %
```

Windows:



```
Command Prompt
C:\Users\Voleti Ravi>python --version
Python 3.9.4
C:\Users\Voleti Ravi>
```

Ubuntu:



```
voletiravi@ubuntu: ~
voletiravi@ubuntu:~$ python3 --version
Python 3.8.5
voletiravi@ubuntu:~$
```

Create a file called requirements.txt with the details below:

```
boto3==1.17.45
botocore==1.20.45
jmespath==0.10.0
python-dateutil==2.8.1
s3transfer==0.3.6
six==1.15.0
urllib3==1.26.4
```

1. Open a terminal
2. `cd` to the directory where `requirements.txt` is located.
3. run:

```
pip3 install -r requirements.txt
```

A terminal window titled "voletiravi -- zsh -- 80x24" showing the output of the command `pip3 install -r requirements.txt`. The output lists the collection and installation of several packages: boto3, botocore, jmespath, python-dateutil, s3transfer, six, and urllib3. A warning message indicates that the user is using pip version 20.2.3, while version 21.0.1 is available, and suggests upgrading via the `'/Library/Frameworks/Python.framework/Versions/3.9/bin/python3.9 -m pip install --upgrade pip'` command.

```
voletiravi@Voletis-MBP ~ % pip3 install -r requirements.txt
Collecting boto3==1.17.45
  Using cached boto3-1.17.45-py2.py3-none-any.whl (131 kB)
Collecting botocore==1.20.45
  Using cached botocore-1.20.45-py2.py3-none-any.whl (7.4 MB)
Collecting jmespath==0.10.0
  Using cached jmespath-0.10.0-py2.py3-none-any.whl (24 kB)
Collecting python-dateutil==2.8.1
  Using cached python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
Collecting s3transfer==0.3.6
  Using cached s3transfer-0.3.6-py2.py3-none-any.whl (73 kB)
Collecting six==1.15.0
  Using cached six-1.15.0-py2.py3-none-any.whl (10 kB)
Collecting urllib3==1.26.4
  Using cached urllib3-1.26.4-py2.py3-none-any.whl (153 kB)
Installing collected packages: jmespath, urllib3, six, python-dateutil, botocore
, s3transfer, boto3
Successfully installed boto3-1.17.45 botocore-1.20.45 jmespath-0.10.0 python-dat
eutil-2.8.1 s3transfer-0.3.6 six-1.15.0 urllib3-1.26.4
WARNING: You are using pip version 20.2.3; however, version 21.0.1 is available.
You should consider upgrading via the '/Library/Frameworks/Python.framework/Vers
ions/3.9/bin/python3.9 -m pip install --upgrade pip' command.
voletiravi@Voletis-MBP ~ %
```

Execution

Set up:

1. script
2. Cron Job

Example of an automatic execution:

```
voletiravi@Voletis-MBP ~ % python3 /Users/voletiravi/PycharmProjects/StorageCalculator/src/lifecycle/lifecycle-current-non-current.py
$ Paginating bucket rv-test-terraform-bucket-minio
-----
$ Before deleting objects
$ current objects: 3
$ non-current objects: 5
-----
$ Deleting objects from bucket rv-test-terraform-bucket-minio
{'ResponseMetadata': {'RequestId': '86D438A294CEC420', 'HostId': 'oQA4ZDRSz4oWFnKy6gETfbnLLI629EgB6a5VjdySM8D02vcYUrKwrTgH2/da6asGcooT7Cbugka', 'HTTPStatusCode': 200, 'HTTPHeaders': {'content-type': 'application/xml', 'date': 'Thu, 08 Apr 2021 19:27:01 GMT', 'server': 'WasabiS3/6.2.4468-2021-03-04-caf9810 (head03)', 'x-amz-id-2': 'oQA4ZDRSz4oWFnKy6gETfbnLLI629EgB6a5VjdySM8D02vcYUrKwrTgH2/da6asGcooT7Cbugka', 'x-amz-request-id': '86D438A294CEC420', 'transfer-encoding': 'chunked'}, 'RetryAttempts': 0}}
$ Paginating bucket rv-test-terraform-bucket-minio
-----
$ After deleting objects
$ current objects: 3
$ non-current objects: 5
-----
$ task complete
voletiravi@Voletis-MBP ~ %
```

Cron Job

Windows

The easiest way to set up a cron job on Windows is by using the Windows Task scheduler. Before we start creating the task schedule itself, we need to get some information to set up tasks. First we need to create a batch file which will execute the python script every day:

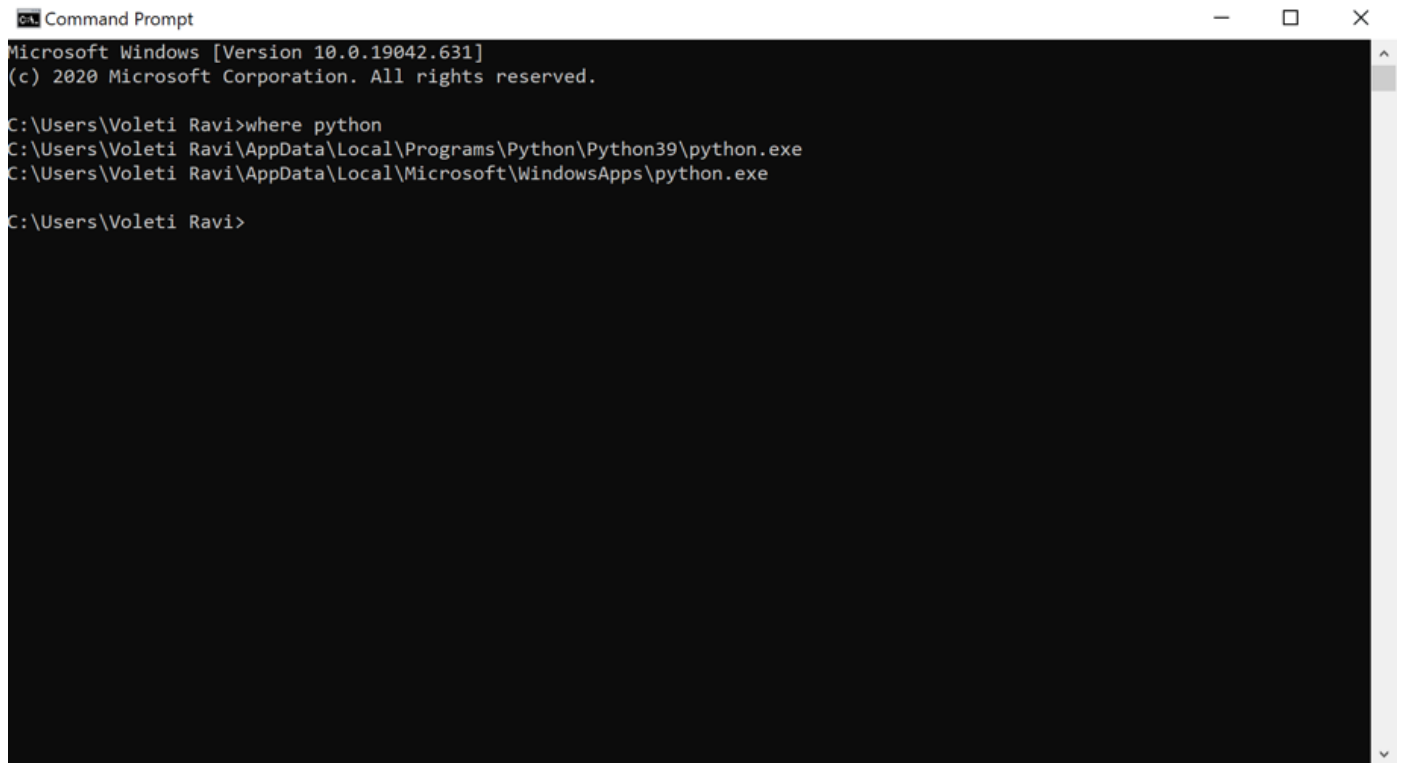
1. create a file named `wasabi-scheduler.bat` To run the python script via the batch script we need to write the following in the file:

```
"Path where your Python exe is stored\python.exe" "Path where your Python script is stored\script name.py"
pause
```

2. In order to do that locate where the python.exe is located. An easy way is to Go to cmd and type:

```
where python
```

It should give you an output like this:



```
Command Prompt
Microsoft Windows [Version 10.0.19042.631]
(c) 2020 Microsoft Corporation. All rights reserved.

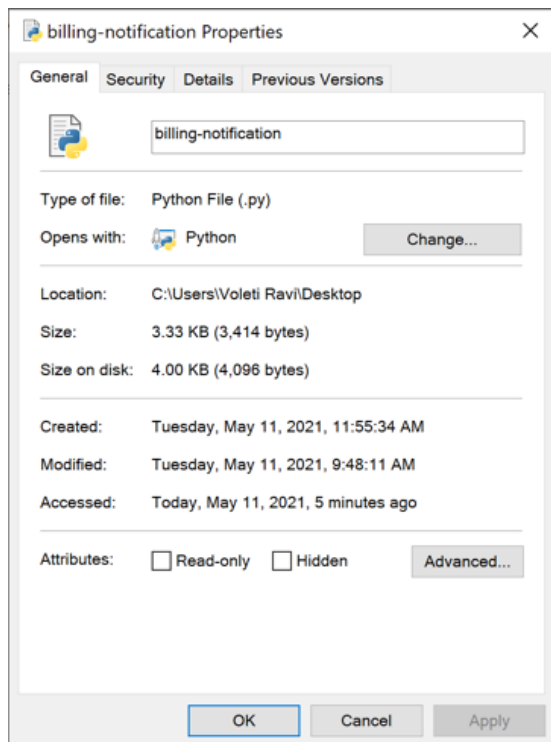
C:\Users\Voleti Ravi>where python
C:\Users\Voleti Ravi\AppData\Local\Programs\Python\Python39\python.exe
C:\Users\Voleti Ravi\AppData\Local\Microsoft\WindowsApps\python.exe

C:\Users\Voleti Ravi>
```

Copy the location of the file and rewrite in the `wasabi-scheduler.bat` file. In my case it is:

```
C:\Users\Voleti Ravi\AppData\Local\Programs\Python\Python39\python.exe
```

3. Next download and copy over the lifecycle script to a preferred location and get its path. An easy way to do that is by right click -> Location.



Copy the location of the file and rewrite in the `wasabi-scheduler.bat` file similar to python location. In my case it is:

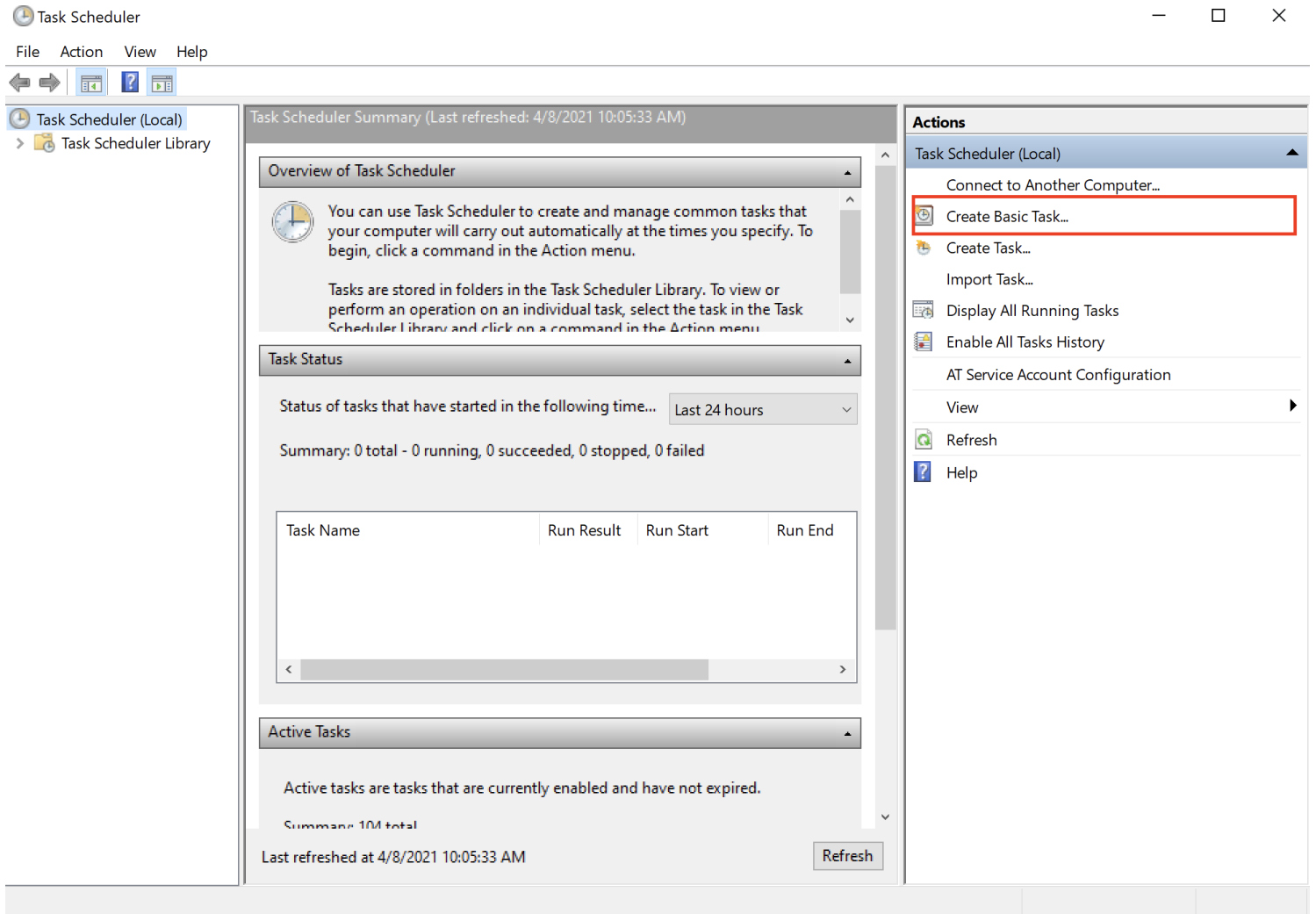
C:\Users\Voleti Ravi\Desktop\billing-notification.py

4. Your batch file should look something like this.

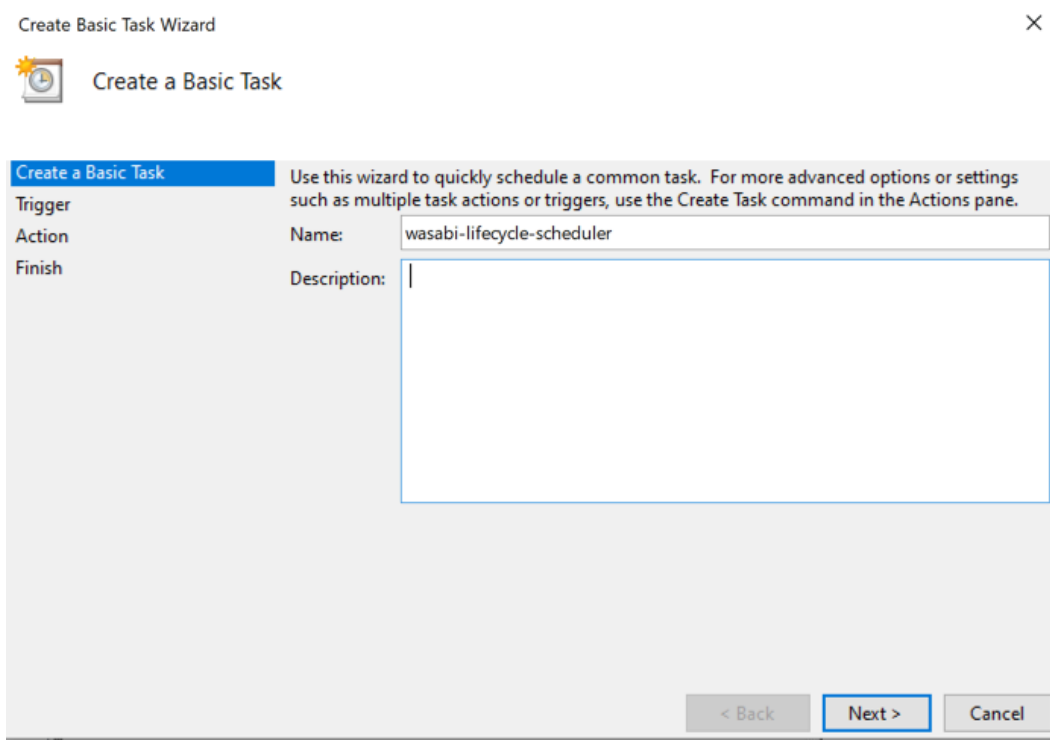
```
"C:\Users\Voleti Ravi\AppData\Local\Programs\Python\Python39\python.exe" "C:\Users\Voleti Ravi\Desktop\billing-not
```

The *pause* feature allows you to not immediately exit the program

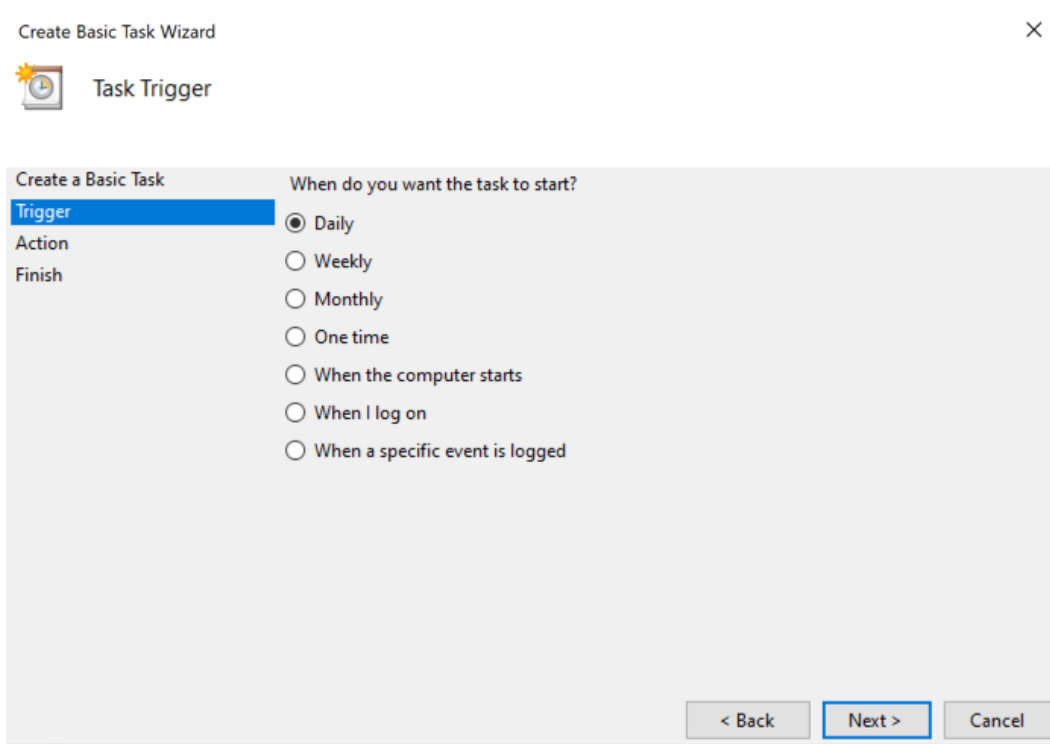
Next Go to Start -> Task Scheduler and create a basic task from the right-hand side.



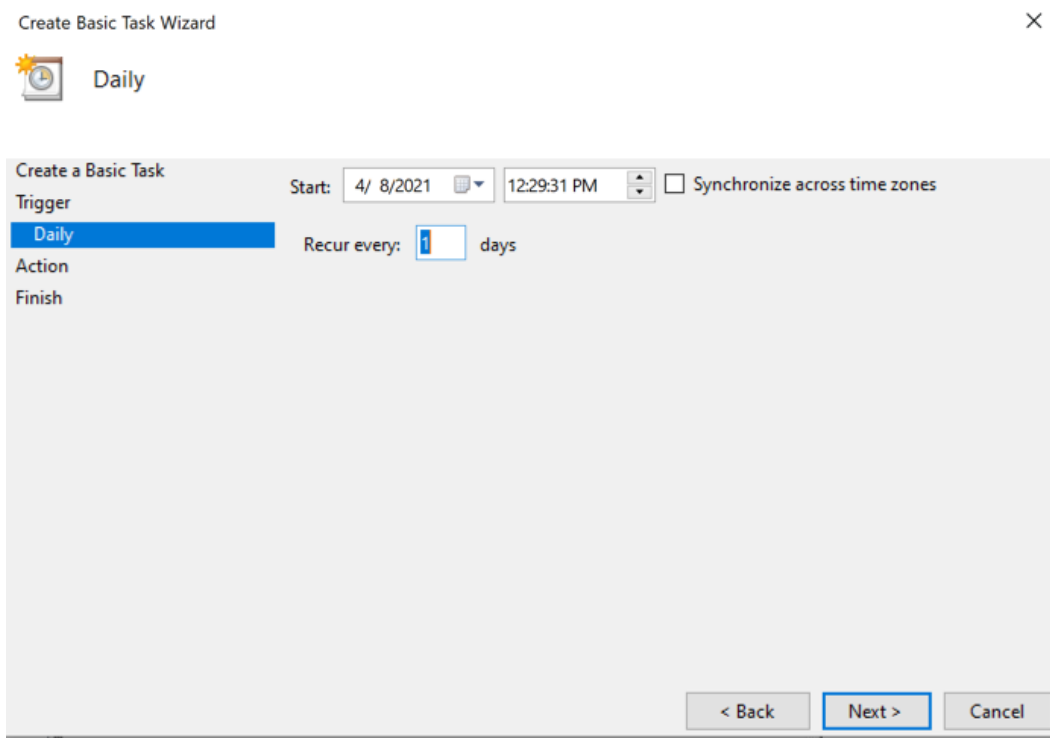
2. Create a name so that you may edit it later, if needed.



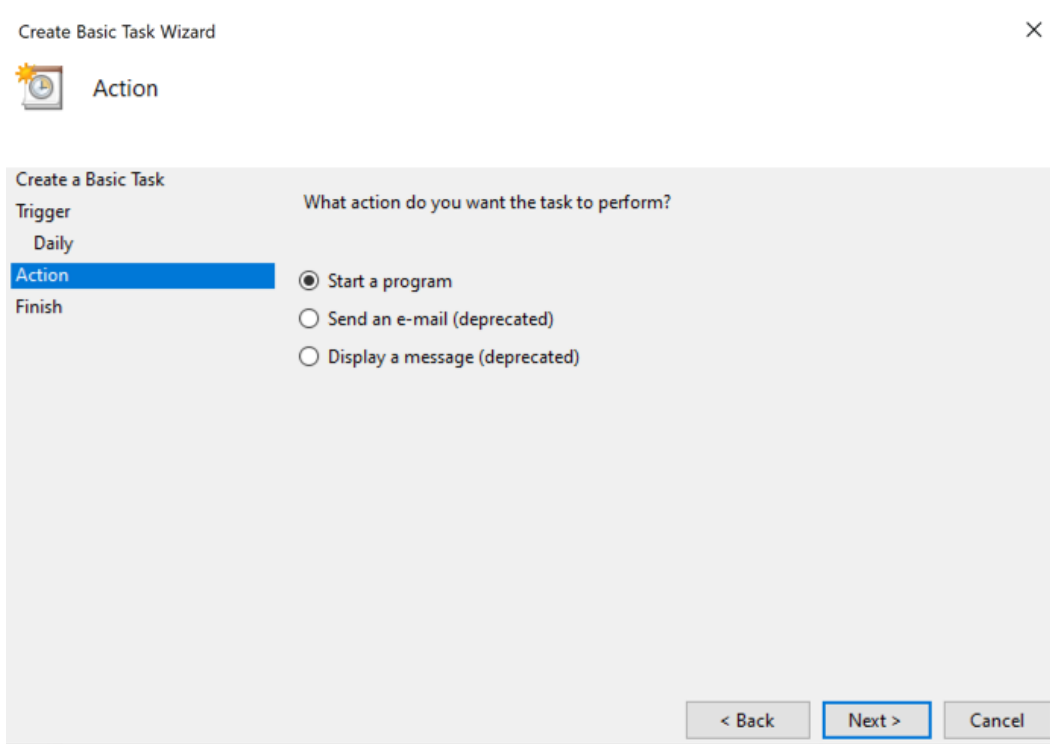
3. Create a daily backup.



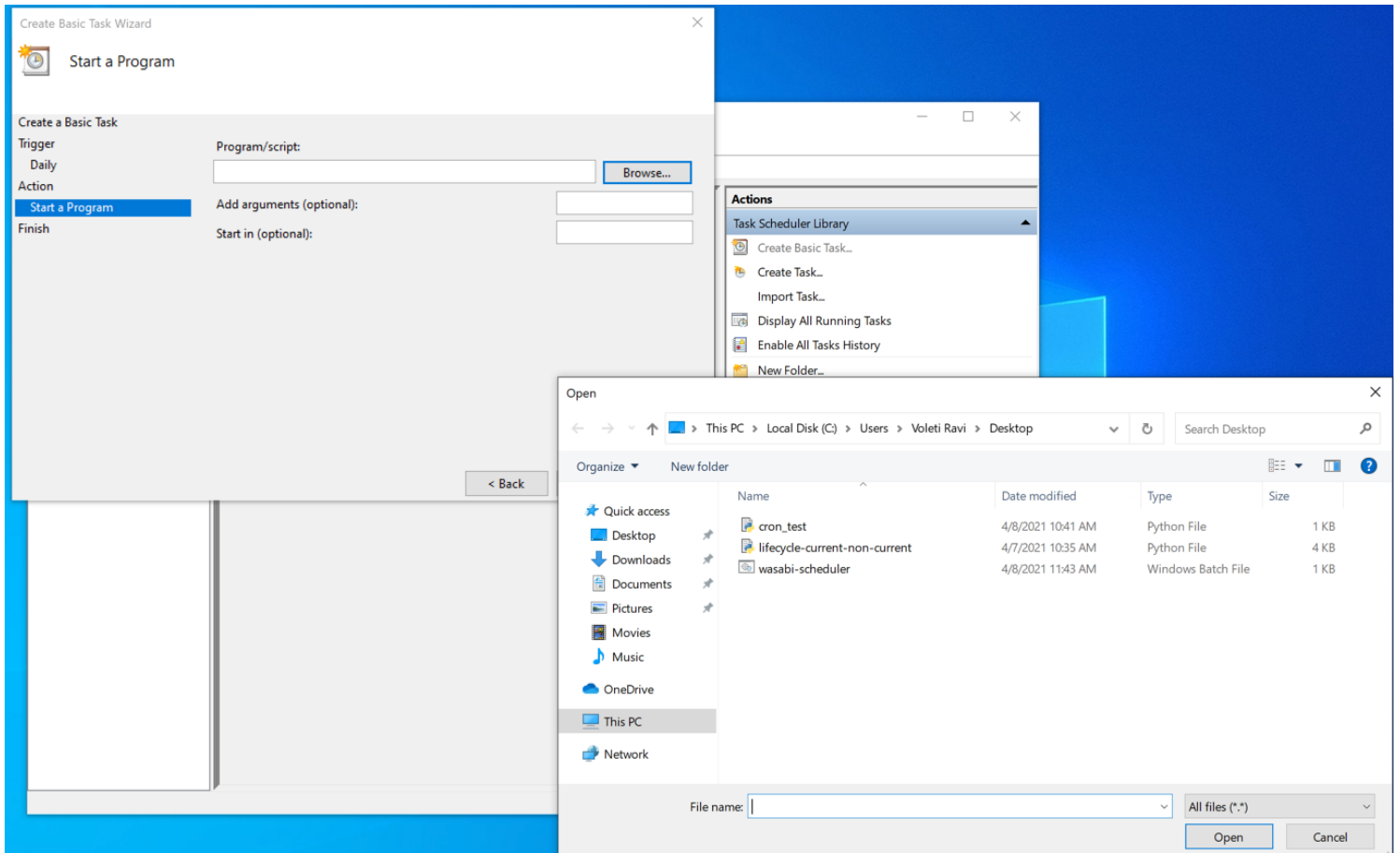
4. Select a time for the task.



5. Select Start a program

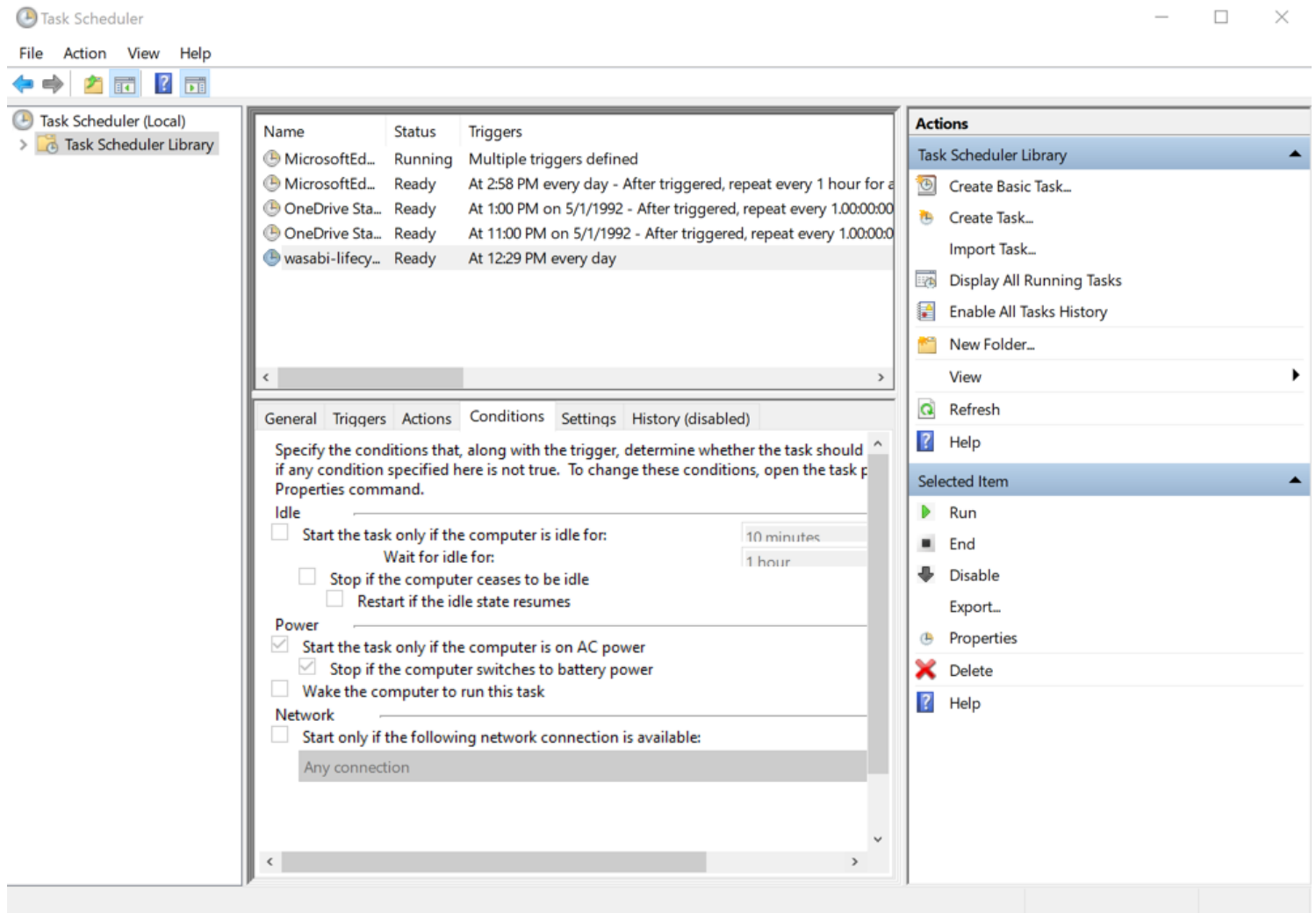


6. Browse and select the wasabi-scheduler.bat file



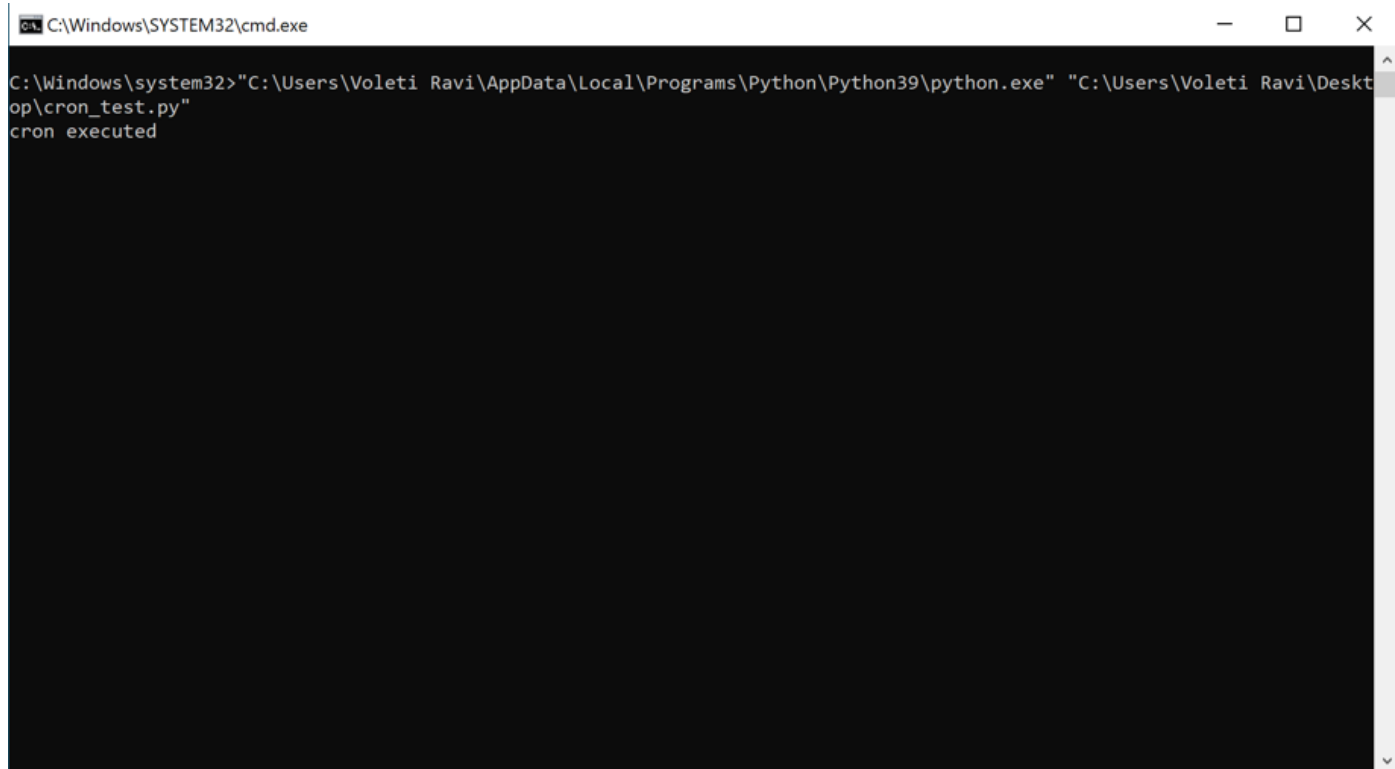
8. Review your changes and press Finish.

You will find the task under Task schedule library. Here you can right click on the task and change any properties.



To test the application you can right click -> press RUN This script should run periodically every day.

Here I am running a dummy script to demonstrate this action.



```
C:\Windows\SYSTEM32\cmd.exe
C:\Windows\system32>"C:\Users\Voleti Ravi\AppData\Local\Programs\Python\Python39\python.exe" "C:\Users\Voleti Ravi\Desktop\cron_test.py"
cron executed
```

To remove you can right click -> delete or select the task and press the delete option.

Mac

The best way to set up cron jobs on Mac is by using `crontab`

Crontab requires setting the time using a specific syntax. You can make use of this website <https://crontab.guru/> to create this syntax.

Please read more about it here: [Crontab quickstart reference](#).

1. you can set up the crontab by typing `crontab -e` in the terminal. This will open a vim console on the terminal:



2. press `i` to get into [insert mode] and paste the follow information:

```
* * * * * osascript -e 'tell app "Terminal" to do script "python3 <absolute path of the python script>" activate'
```

NOTE:

Remember to replace the * * * * * with the correct time syntax.

Reason for: `osascript -e 'tell app "Terminal" to do script "python3 <absolute path of the python script>" activate` Crontab runs tasks in the background without any visuals. In order to see an output You can use AppleScripts combined with cron in order to emulate opening a Terminal and running the script from within the Terminal.

if you wish to not do that you can simply run:

```
* * * * * /usr/bin/python3 <absolute path of the python script>
```

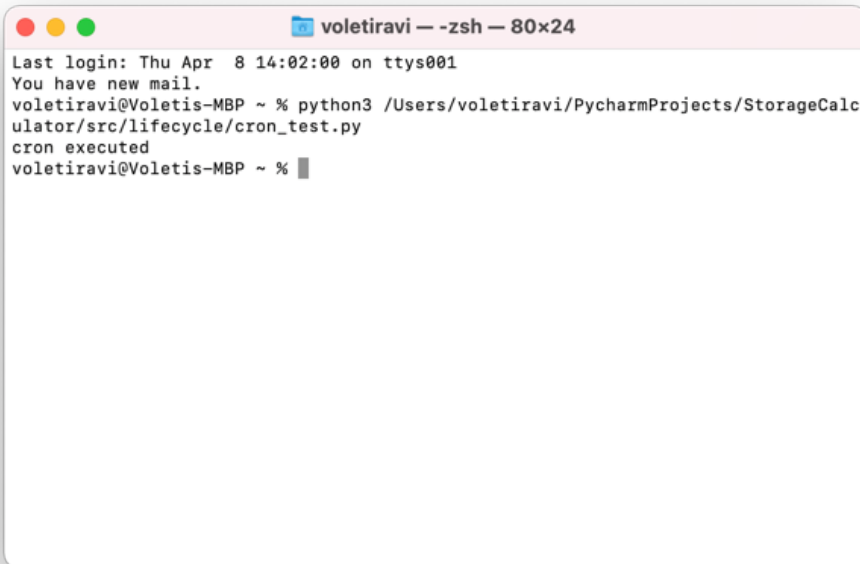
In our execution this is how it looks like:

```
* * * * * osascript -e 'tell app "Terminal" to do script "python3 /Users/voletiravi/PycharmProjects/StorageCalcula
```

3. Once you have done this, press ESC to exit Insert mode. Then type ":wq" to save your changes.



4. The script we just demonstrated will run the Cronjob every 1 minute.

A terminal window titled "voletiravi — zsh — 80x24" showing the execution of a cron job. The output includes a login message, a notification of new mail, the execution of a Python script, and a confirmation that the cron job was executed.

```
voletiravi@voletis-MBP ~ % python3 /Users/voletiravi/PycharmProjects/StorageCalculator/src/lifecycle/cron_test.py
cron executed
voletiravi@voletis-MBP ~ %
```

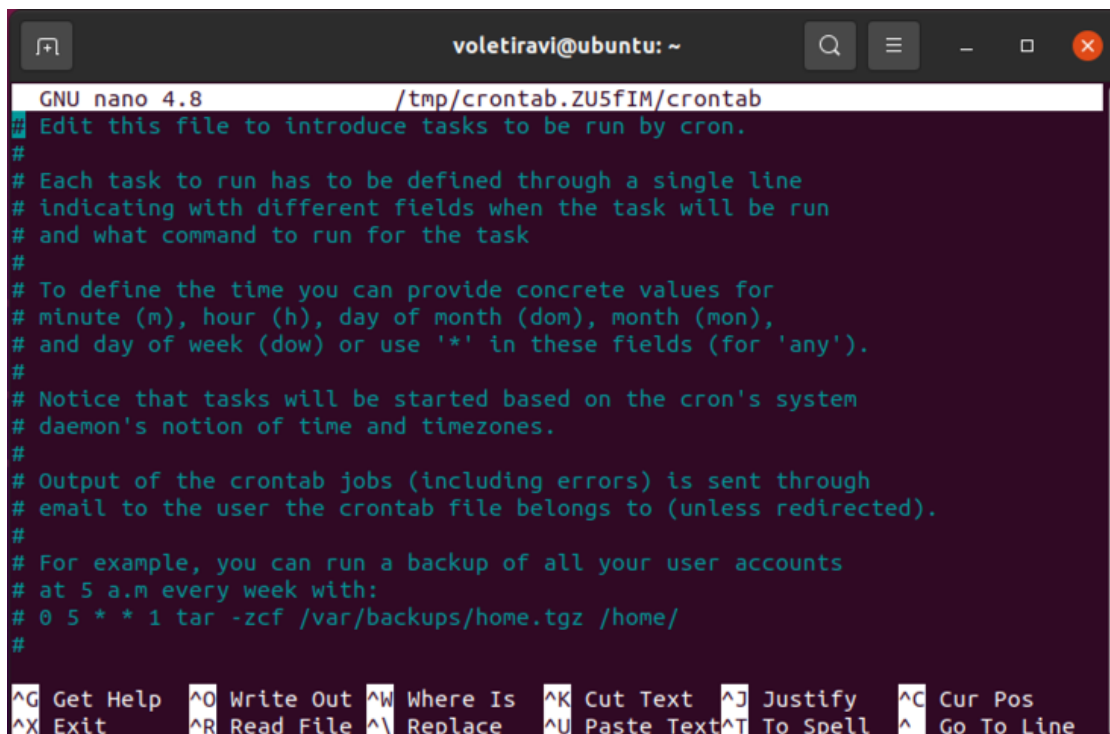
5. To remove the cronjob. Type `crontab -e`, press `i` to go into [insert mode] remove the entry and press `escape` and `:wq` to quit.

Ubuntu

The best way to set up cron jobs on Ubuntu is by using `crontab`

Crontab requires setting the time using a specific syntax. You can make use of this website <https://crontab.guru/> to create this syntax. Please read more about it here: [Crontab quickstart reference](#).

1. you can set up the crontab by typing `crontab -e` in the terminal. This will open a console on the terminal with your preferred editor, we are demonstrating this with nano:

A terminal window showing the nano editor editing a crontab file. The editor displays instructions on how to define cron jobs, including fields for minute, hour, day of month, month, and day of week, and an example of a backup task.

```
GNU nano 4.8 /tmp/crontab.ZU5fIM/crontab
Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line
```

2. scroll down to the bottom of the file and paste the follow information:

```
* * * * * /usr/bin/python3 <absolute path of the python script>
```

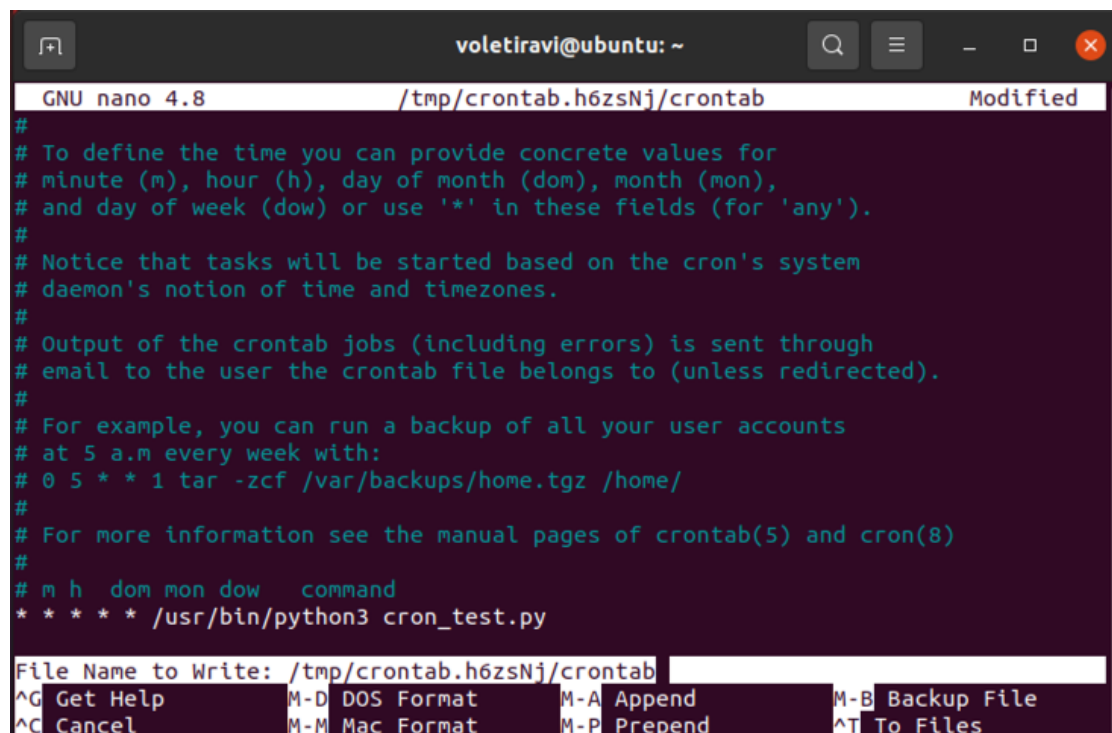
NOTE:

Remember to replace the * * * * * with the correct time syntax.

In our execution this is how it looks like:

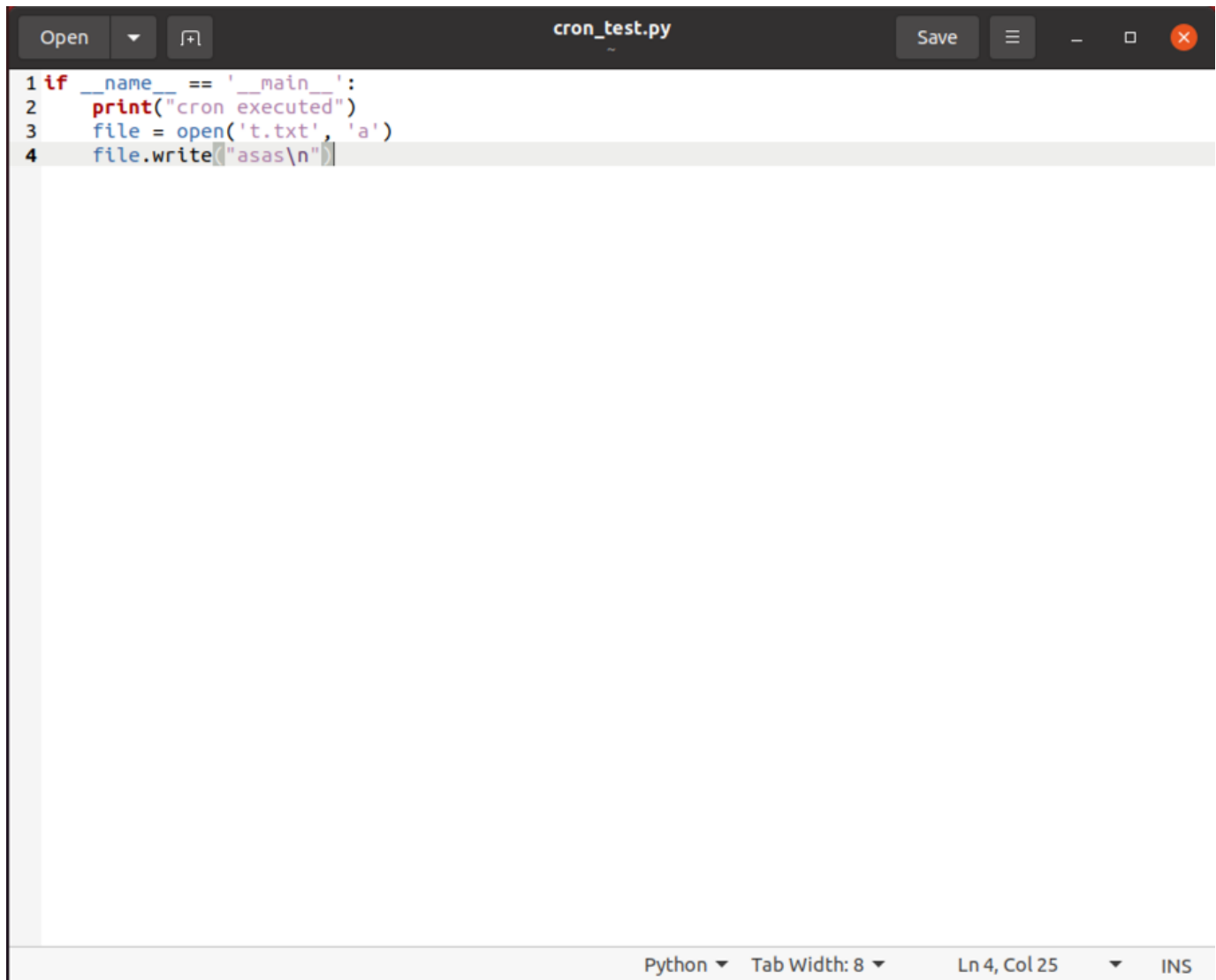
```
* * * * * /usr/bin/python3 cron_test.py
```

3. Once you have done this press ^X to exit.



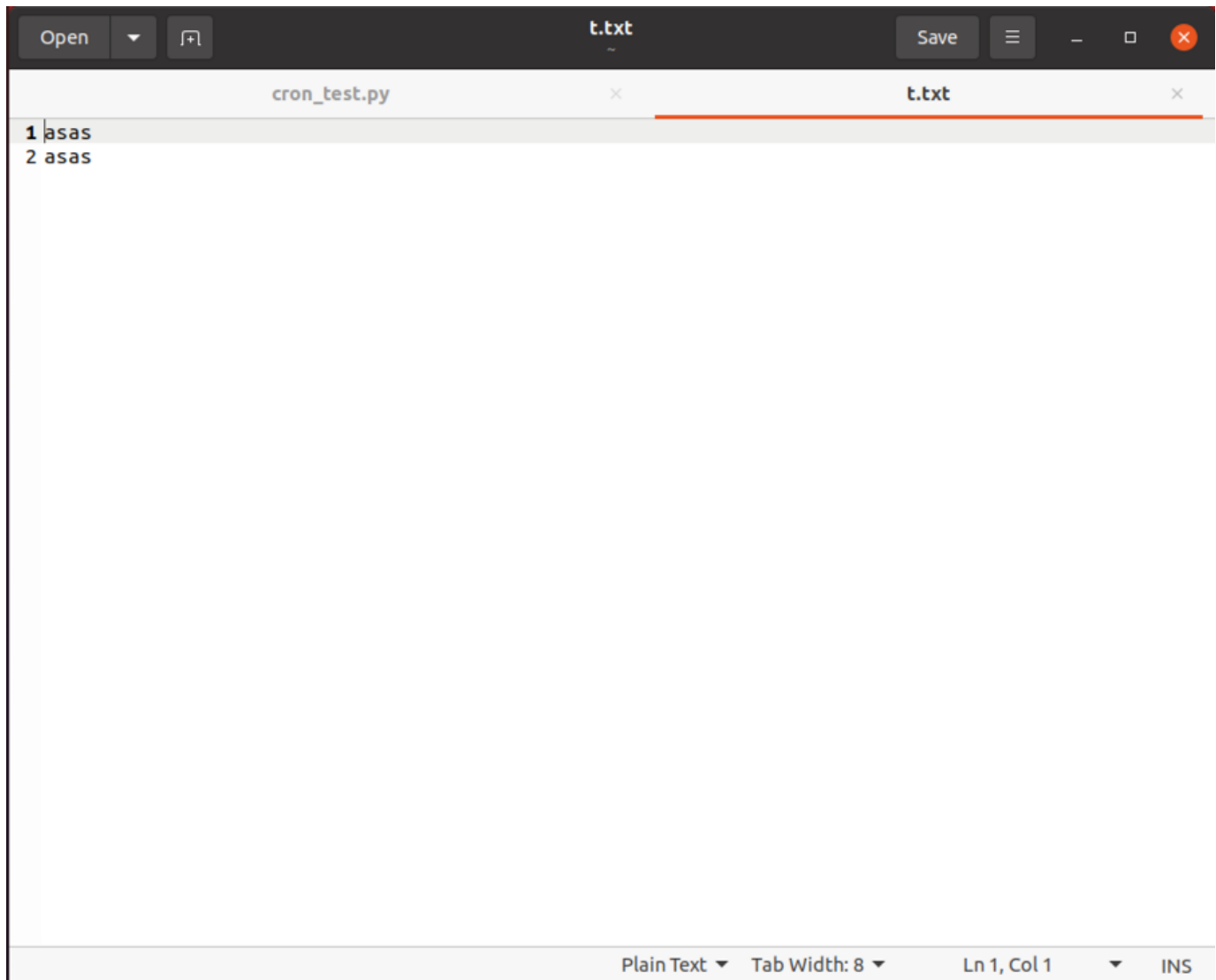
```
GNU nano 4.8 /tmp/crontab.h6zsNj/crontab Modified
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
* * * * * /usr/bin/python3 cron_test.py
File Name to Write: /tmp/crontab.h6zsNj/crontab
^G Get Help      M-D DOS Format  M-A Append     M-B Backup File
^C Cancel        M-M Mac Format  M-P Prepend    ^T To Files
```

4. The script we just demonstrated will run the Cronjob every 1 minute. This particular script writes data to a text file `t.txt`



```
1 if __name__ == '__main__':
2     print("cron executed")
3     file = open('t.txt', 'a')
4     file.write("asas\n")
```

Python ▾ Tab Width: 8 ▾ Ln 4, Col 25 ▾ INS



```
1 | asas
2 | asas
```

5. To remove the cronjob. Type `crontab -e`, press `i` to go into insert mode remove the entry and press `escape` and `:wq` to quit.