



THREAD thread_qDhtW65TfNJ2CyoainEAhtxB

Run

Clear

```
pip install requests
pip install smtplib
```

After that, you can use the following code to get the status continuously and send an email when the status is `free`.

```
import time
import requests
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

# Assuming arbin status API returns a json with status in 'status' field
def check_status(port, channel):
    status = arbin.status(port, channel)
    return status['status']

def send_email():
    mail_content = "Machine is free now"
    #The mail addresses and password
    sender_address = 'abc@gmail.com'
    sender_pass = 'password'
    receiver_address = 'def@gmail.com'

    #Setup the MIME
    message = MIMEMultipart()
    message['From'] = sender_address
    message['To'] = receiver_address
    message['Subject'] = 'Machine Status Update' #The subject line

    #The body and the attachments for the mail
```



```
        session.quit()
    )
    print('Mail Sent')
)
def main():
    port = 'PortNumber' # Update port
    channel = 'ChannelNumber' # Update channel
    while True:
        status = check_status(port, channel)
        if status == 'free':
            send_email()
            break
        time.sleep(900) # Pauses execution for 900 seconds (15 minutes)
)
if __name__ == '__main__':
    main()
```

Remember to replace 'PortNumber' and 'ChannelNumber' with your port and channel . Also, replace 'abc@gmail.com', 'def@gmail.com', and 'password' with your sender's email, receiver's email, and sender's email password respectively.

Note: If you are using Gmail as your SMTP service, you may need to configure "Less Secure App" settings.

User

That is great. Now can you modify the script so that the function input in check_status can be received through a command line/another python script

software developer

Absolutely. We can use the argparse package, which is a standard Python library, for command-line parsing. Here's how we can modify the main() function and the script:

```
1 import argparse
2
3 def parse_args():
4     parser = argparse.ArgumentParser(description='Check Arbin machine
```



```
15         print("Machine is now free. Sending email ...")
16         send_email()
17         break
18     print("Machine is busy. Checking again in 15 minutes.")
19     time.sleep(900) # Pauses the program for 900 seconds (15 minutes)
20
21 if __name__ == '__main__':
22     main()
```

Now you can run the script from the command-line or another python script by providing the `port` and `channel` as arguments. Here's an example of how you can run it:

```
python script.py --port PortNumber --channel ChannelNumber
```

Just replace `PortNumber` and `ChannelNumber` with your `port` and `channel`.

User

This works. Now I would like the script run at background so that when it keep checking the status it would not affect other functions/scripts running at the same time

software developer

The easiest way to run a Python script in the background is to use the `&` command in the terminal:

```
python script.py --port PortNumber --channel ChannelNumber &
```

The `&` makes the command run in the background.

If you want to keep the script running even after closing the session/terminal, you can use `nohup`:

```
nohup python script.py --port PortNumber --channel ChannelNumber &
```



User

I can not use command line to do this, is there a way for python code make it run in background? This use case is part of a web app, so I would like the user do other stuff in the web app when it keeps checking the machine status.

software developer

In a web-app scenario, the best practice is to use a task queue or background job architecture. Python's popular background task processing libraries include `Celery`, `RQ (Redis Queue)`, and others.

However, if your use case is not complex and you want no third-party dependencies, you can use Python's built-in `threading` or `multiprocessing` library to achieve this order. If you use `threading`, note that it is not suitable for CPU-bound tasks due to the Global Interpreter Lock (GIL) in Python. If CPU-intensive tasks must be performed, `multiprocessing` is recommended.

Below is an example of how you might use `multiprocessing`:

```
1  import multiprocessing
2  import argparse
3
4  def parse_args():
5      parser = argparse.ArgumentParser(description='Check Arbin machine
6      parser.add_argument('--port', required=True, help='Port of machine
7      parser.add_argument('--channel', required=True, help='Channel of m
8      return parser.parse_args()
9
10 def run_script(port, channel):
11     while True:
12         status = check_status(port, channel)
13         if status == 'free':
14             print("Machine is now free. Sending email ...")
15             send_email()
16             break
17     print("Machine is busy. Checking again in 15 minutes.")
18     time.sleep(900) # Pauses the program for 900 seconds (15 minut
```





Please note that you should have error handling in your child process because if it crashes, the main process may not be aware of it.

However, for production Web Apps, I would recommend using a task queuing system as it offers more control and robustness. Examples are `Celery` with a message broker like `RabbitMQ` or `Redis`, `RQ (Redis Queue)`, `Huey`, etc.

User

Thanks for the recommendation. Can you give me an example code to use a task queuing system to meet my goal?

software developer

Sure, I'm going to use `Celery` as a task queue with `RabbitMQ` as a broker, and `Redis` for results backend. `Celery` is a robust and effective way to handle async tasks and distributed tasks in Python.

- 1 First, you need to install RabbitMQ and Redis. They are available on most Unix-like systems via package manager. On Ubuntu, you would install them with:

```
sudo apt-get install rabbitmq
sudo apt-get install redis-server
```

You also need to install Celery in your Python environment:

```
pip install Celery
pip install redis
```

Then, define a Celery worker that will check the status every 15 minutes and send an email when the machine is free:

```
1 from celery import Celery
2
3 app = Celery('tasks', broker='pyamqp://guest@localhost//', backend='r
4
5 @app.task
```





```
1 import tasks
2
3 result = tasks.check_status_send_email.delay(port, channel)
```

- 4 Start the Celery worker from command line:

```
celery -A tasks worker --loglevel=info
```

The script now runs asynchronously and the task id is in the `result` object; You can check the task status using `result.status` and get the return value using `result.result`.

Note: You'll need to modify to suit your needs in terms of connection information and configuration for RabbitMQ and Redis.

User

Can you explain to me more about how to use RabbitMQ and Redis? Say I'm running my Web app on local host and AWS cloud this two cases

software developer

Sure, `RabbitMQ` and `Redis` are two separate technologies used for different purposes within the Celery architecture.

- 1 `RabbitMQ` is a message broker. It's responsible for coordinating and managing communication between distributed systems - in this case, it's handling the delivery of task messages to one or multiple Celery workers. The message broker can be thought of as a post office: it helps route your messages (tasks) to the correct recipient (worker).

To install `RabbitMQ` locally, you would typically use a package manager. For Ubuntu, the command is: `sudo apt-get install rabbitmq-server`. On AWS, you can use the Amazon MQ service, which is a managed message broker service for RabbitMQ.

`Redis` is an in-memory data structure store, used as a database, cache, and task-queue. In a Celery setup, it's used to store task results (used as Celery's backend).





If you have Docker installed, both RabbitMQ and Redis can be started as Docker containers which makes it simpler to control and manage. Just pull their respective images from Docker hub and run them.

User

If I'm running my web app by python flask/django, how can I start RabbitMQ and Redis?

software developer

Whether you're running a Flask or Django app doesn't affect starting RabbitMQ or Redis. These services run independently of your app and have their own setup procedures.

If you're running your app on your local system you would start RabbitMQ and Redis via system services commands.

To start RabbitMQ:

```
sudo service rabbitmq-server start
```

To start Redis:

```
sudo service redis-server start
```

Once these services are running, your Flask/Django app can interact with them via Celery. In a production environment or for testing in an environment similar to production, you could use Docker to manage these services along with your Django or Flask app.

You can define a `docker-compose.yml` file like this:

```
1  version: '3'
2  services:
3    rabbit:
4      image: "rabbitmq"
5      ports:
6        - "5672:5672"
7    redis:
8      image: "redis"
9    web:
```





This docker-compose file set up a RabbitMQ service, a Redis service and a Flask app service altogether. When you run `docker-compose up`, Docker starts all these services, the Flask app will be able to interact with RabbitMQ and Redis via the network automatically set up by Docker. Just remember to replace `flask run --host=0.0.0.0` with the command you use to start your Flask/Django server.

To call a Celery task in your app, it's as simple as importing the task function and using `.delay()` to add the task to the task queue.

```
from your_celery_tasks_module import your_task
your_task.delay(arg1, arg2)
```

When called with `delay` or `apply_async`, the task is processed asynchronously, either in the same process (for `CELERY_ALWAYS_EAGER=True`) or in the worker processes.

User

What if my local system is windows? What would be the command line code to start RabbitMQ and Redis

software developer

On a Windows system, starting RabbitMQ and Redis involves a different process compared to Unix-based systems.

RabbitMQ:

First, install Erlang because RabbitMQ requires the Erlang runtime. Then install RabbitMQ. Both Erlang and RabbitMQ are available as .exe files for Windows, which you can run directly:

- 1 Install Erlang: <https://www.erlang.org/downloads>
- 2 Install RabbitMQ: <https://www.rabbitmq.com/install-windows.html>

After installation, RabbitMQ server can be started in the "RabbitMQ Command Prompt" via `rabbitmq-server` command. To start it as a service, you can run:

```
rabbitmq-service start
```

Redis:





like systems. If you have to use Windows, Windows Subsystem for Linux (WSL) is a good alternative.

User

Thanks. How can I save our today's chat history?

software developer

If you're communicating via a platform like OpenAI's chat interface, you can usually save the chat history by simply copying the text and pasting it into a document for safekeeping. This would depend on the platform you're using to communicate with the AI.

Some platforms may also have a built-in feature for exporting or downloading chat history.

You'd have to check the specific features of the platform you're using.

Lastly, if all else fails, you can take screenshots of the chat and save those for future reference.

Enter your message...

 Add and run Add 

Playground messages can be viewed by anyone at your organization using the API.

