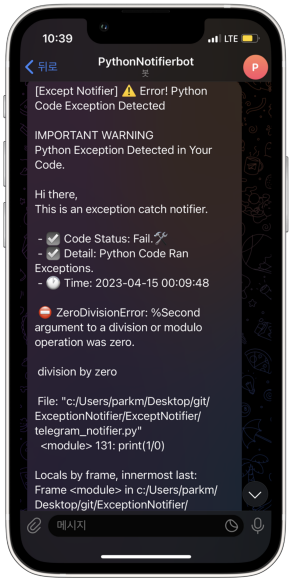
Development Status :: 2 - Pre-Alpha

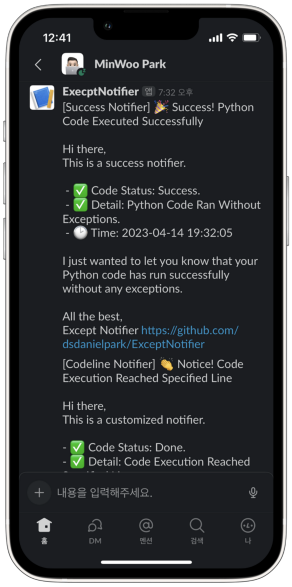
Before QA

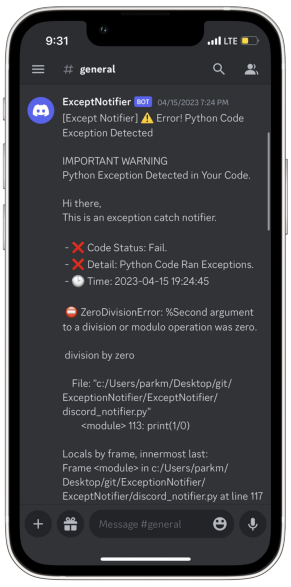
Except Notif!er

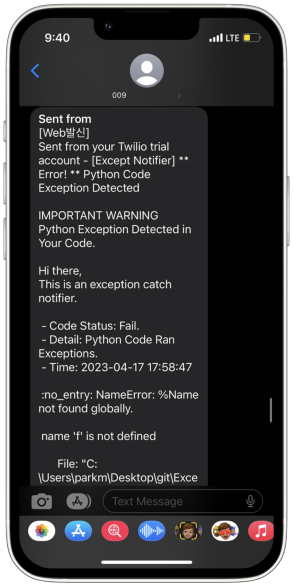
Integrates AI-assisted debugging notifications into Python try-except statements for various messaging applications.

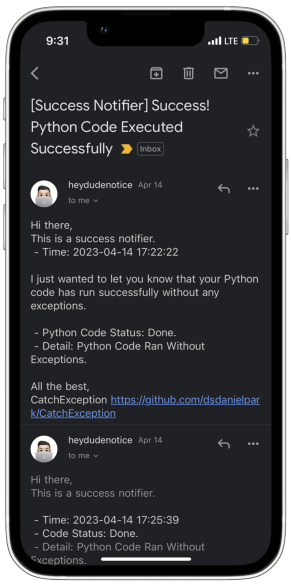
pypi ExceptNotifier | pypi v0.2.0 | downloads 4k | code style black









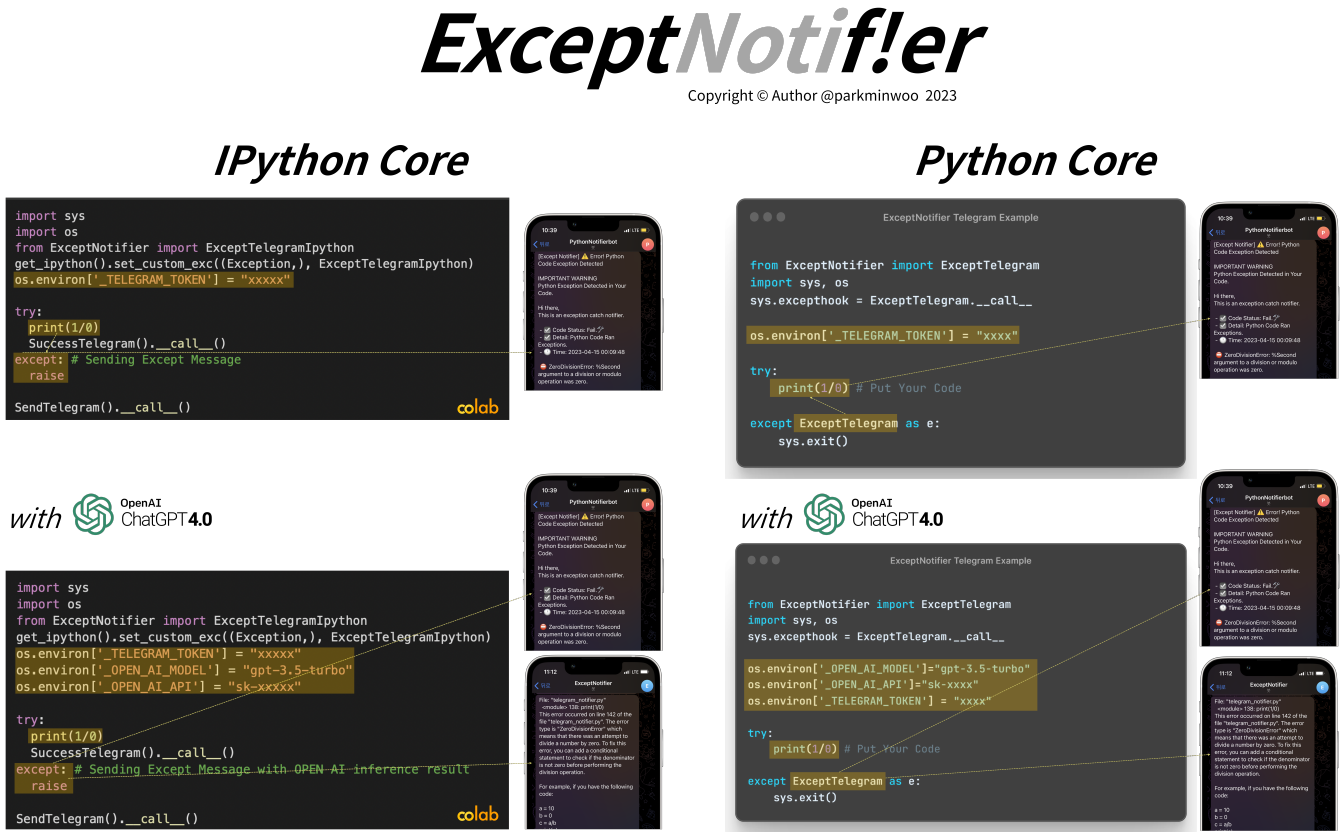


Provides a notification from the application shown in the captured screen.

Python Package: ExceptNotifier

The `ExceptNotifier` Python package offers a flexible approach to receiving notifications by enhancing Python's try-except statement. This package enables you to receive alerts through various messaging applications or emails.

With `ExceptNotifier`, you can obtain detailed compilation errors, including debug information, sent directly to your preferred messaging platform or email. By integrating OpenAI's ChatGPT, you can receive additional error code information as long as you provide the required API model name and key. This feature ensures that error handling and notifications are more informative and accessible, streamlining your debugging process.



Supporting Applications

Applicable to both IPython and Python, but needs to be ported differently only in `ExceptNotifier`. Please refer to the detailed example. (`SuccessNotifier` and `SendNotifier` have the same syntax.)

- Telegram
- Discord
- Slack
- Google Mail
- Line
- AWS Chime
- Microsoft Teams
- Kakao Talk

- Wecaht
- SMS Sending using Twilio
- Desktop Notification using Plyer
- Beep Sound from system
- Opea AI API - If you have OpenAI API Key and model name, you can get information and code examples for debugging in any application.

Quick Start

```
$ pip install ExceptNotifier
```

```
$ pip install exceptnotifier
```

Contents

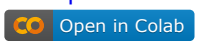
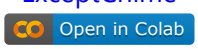
- App Setup Overview
- Tutorial
- Python Core
    - ExceptNotifier
        - AI Debbugging Infomation Notification
    - SuccessNotifier
    - SendNotifier
    - Sender
- IPython Core
    - ExceptNotifier in Ipython
    - SuccessNotifier in Ipython
    - SendNotifier in Ipython
    - Sender in Ipython
- Applied in each application
    - *Telegram*
        - a. Notifier without OpenAI API
        - b. Notifier with OpenAI API
    - *Mail*

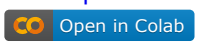
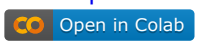
    - *Discord*
    - *Chime*
    - *Slack*
    - *Line*
    - *SMS*
    - *Teams*
    - *Kakaotalk*
    - *Wechat*
    - *Beep*
    - *Desktop*

- Contributing Guide
- License
- Code of Conduct
- Contacts

App Setup Overview

- The variables in the following table must not be contaminated.
- Depending on the situation, consider designating them as global variables for use.
- As you already know, API Keys or security tokens must be secured. Note that the key values which exposured in github will be expired after insecured.
- We are trying to maintain the current architecture as much as possible by considering various methods such as inheriting decorators and Excepthook. We set it as an environment variable as follows, and we are refactoring and testing for a better method.

App	Required Enviroment Variables	Free or Paid	Ease of Setup	Time Required for Setup	Guide Tutorial Link
Beep	N/A	Free	N/A	0min	ExceptBeep
Desktop	N/A	Free	N/A	0min	ExceptDesktop
Telegram	`_TELEGRAM_TOKEN`	Freemium	Easy	2min	ExceptTelegram 
Discord	`_DISCORD_WEBHOOK_URL`	Freemium	Easy	1min	ExceptDiscord 
AWS Chime	`_CHIME_WEBHOOK_URL`	Freemium	Easy	1min	ExceptChime 

App	Required Enviroment Variables	Free or Paid	Ease of Setup	Time Required for Setup	Guide Tutorial Link
Slack	`_SLACK_WEBHOOK_URL`	Freemium	Easy	3min	ExceptSlack 
G-Mail	`_GAMIL_RECIPIENT_ADDR`, `_GMAIL_SENDER_ADDR`, `_GMAIL_APP_PASSWORD_OF_SENDER`	Restricted free	Medium	3min	ExceptMail
Line	`_LINE_NOTIFY_API_TOKEN`	Freemium	Medium	4min	ExceptLine 
SMS	`_TWILIO_SID`, `_TWILIO_TOKEN`, `_RECIPIENT_PHONE_NUMBER`, `_SENDER_PHONE_NUMBER`	Not free	Medium	5min	ExceptSMS
Microsoft Teams	`_TEAMS_WEBHOOK_URL`	Not Free	Medium	5min	ExceptTeams
KakaoTalk	`_KAKAO_TOKEN_PATH`	Freemium	Hell	>=10min(Token refreshes daily)	ExceptKakao

If you add the following two variables to the required variables for each application in the table above, you can receive error location and explanation, as well as examples, from OpenAI's model

API	Required Variables	Free or Paid	Ease of Setup	Time Required for Setup	Guide Tutorial Link
OpenAI API	`Required variables for each application`+ `_OPEN_AI_MODEL`,`_OPEN_AI_API`	Not free	Easy	2min	APIOpenAI

Tutorial

I will update tutorial ASAP. `README.md` is sufficient, but read the application's official documentation if necessary. However, we are preparing a more detailed and friendly tutorial.

1. Main-tutorials: Notebook
2. Sub-tutorial-folder: Tutorials for each function can be found in this folder. The tutorial is synchronized with the Python file name provided by ExceptNotifier.

In this example, some API keys were exposed by creating and removing a test application, but for security reasons, your API key should not be exposed to the outside world.

Python Core

To use the desired application, you must define the necessary variables. Ensure that the variable names remain unchanged, and you can use either local or global variables. If you are using `Telegram`, an example is attached as an image.

Except`Notifier`

If you use Python's try except statement as it is, but change except as follows, you can receive notifications through your application.

- Format: Except`[appName]`
- Type: class *ExampleClass*

```
ExceptChime, ExceptTelegram, ExceptDiscord, ExceptSMS, ExceptMail, ExceptKakao,
ExceptLine, ExceptSlack, ExceptTeams, ExceptDesktope, ExceptBeep
```

Example

```
import sys, os
from ExceptNotifier import ExceptTelegram
sys.excepthook = ExceptTelegram.__call__

os.environ['_TELEGRAM_TOKEN'] = "xxxx"

try:
    print(1/0)
except ExceptTelegram:    # sending except message to telegram
    sys.exit()
```

AI Debbugging Infomation Notification

You can receive debugging information from ChatGPT via OpenAI's API when using the Except statement. The syntax remains the same, but you'll need to configure these two variables: `_OPEN_AI_MODEL`,`_OPEN_AI_API`

Example

```
import sys, os
from ExceptNotifier import ExceptTelegram
sys.excepthook = ExceptTelegram.__call__

os.envirion['_TELEGRAM_TOKEN'] = "xxxx"
os.envirion['_OPEN_AI_MODEL']="gpt-3.5-turbo"
os.envirion['_OPEN_AI_API']="sk-xxxxxx"

try:
    print(1/0)
```

```
except ExceptTelegram: # sending msg WITH AI DEBUGGING to telegram
    sys.exit()
```

SuccessNotifier

- Format: Success[appName]
- Type: Class
  ExampleClass
  By placing the try except in python at the end of the try statement, applications can be notified that the try statement worked normally.

```
SuccessChime, SuccessTelegram, SuccessDiscord, SuccessSMS, SuccessMail,
SuccessKakao, SuccessLine, SuccessSlack, SuccessTeams, SuccessDesktope, SuccessBeep
```

Example

```
import sys, os
from ExceptNotifier import SuccessTelgeram
sys.excepthook = ExceptTelegram.__call__
os.environ['_TELEGRAM_TOKEN'] = "xxxx"

try:
    print(1/20)
    SuccessTelgeram().__call__()  # sending success message to telegram
except:
    sys.exit()
```

SendNotifier

- Format: Send[appName]
- Type: class
  ExampleClass
  Place it anywhere on the line of code you want, and you'll be notified when that line of code is reached.

```
SendChime, SendTelegram, SendDiscord, SendSMS, SendMail, SendKakao, SendLine,
SendSlack, SendTeams, SendDesktope, SendBeep
```

Example

```
import sys, os
from ExceptNotifier import SendTelegram
```

```
sys.excepthook = ExceptTelegram.__call__
os.environ['_TELEGRAM_TOKEN'] = "xxxx"

SendTelegram().__call__() # sending message to telegram

noti = SendTelegram()
noti()                    # sending message to telegram
```

Sender

It is recommended to conduct a simple message sending test through the `sender`. Assuming that you can communicate with REST API or WEBHOOK normally, `ExceptNotifier` can work normally.

- Every application's ExceptNotifier uses the sender method.
- Format: send_`[appName]`_msg
- Type: Function
  Example

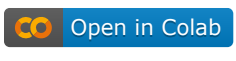
```
send_chime_msg, send_telegram_msg, send_discord_msg, send_sms_msg, send_gmail_msg,
send_kakao_msg, send_line_msg, send_slack_msg, send_teams_msg, send_desktop_msg,
beep
```

Example

```
from ExceptNotifier import send_telegram_msg

send_telegram_msg(_TELEGRAM_TOKEN, "Any Test Message")
```

IPython Core

You can use all the same except for the python code and `ExceptNotifier` mentioned above. In other words, the `SuccesNotifier`, `SendNotifier`, and `Sender` functions can all be used same in IPython without any special processing. Only `ExceptNotifier` is need to be defined.

- Example in Telegram 

Except`Notifier` in Ipython

You have to use `raise` in Ipython ExceptNotifier.

- Format: Except[appName]
- Type: function

Example code without Open AI API

```
from ExceptNotifier import ExceptTelegramIpython
import os
os.environ['_TELEGRAM_TOKEN'] = "xxxxx"
get_ipython().set_custom_exc((Exception,), ExceptTelegramIpython)

try:
  print(1/0)
except:
  raise
```

Example code With Open AI API

```
from ExceptNotifier import ExceptTelegramIpython
import os
get_ipython().set_custom_exc((Exception,), ExceptTelegramIpython)

os.environ['_OPEN_AI_MODEL'] = "gpt-3.5-turbo"
os.environ['_OPEN_AI_API'] = "sk-xxxxx"
os.environ['_TELEGRAM_TOKEN'] = "xxxxx"

try:
  print(1/0)
except:
  raise
```

SuccessNotifier in Ipython

- Same syntax in Python. See Python Core above.

```
import sys
import os
from ExceptNotifier import SuccessTelegram
os.environ['_TELEGRAM_TOKEN'] = "xxxxx"

try:
  print(1/20) # Your Code Here
  SuccessTelegram().__call__()    # Sending Success message
except:
  raise
```

Send`Notifier` in Ipython

- Same syntax in Python. See Python Core above.

```
import sys
import os
from ExceptNotifier import SendTelegram
os.environ['_TELEGRAM_TOKEN'] = "xxxxx"

SendTelegram().__call__()         # Sending telegram message
```

Sender in Ipython

- Same syntax in Python. See Python Core above.

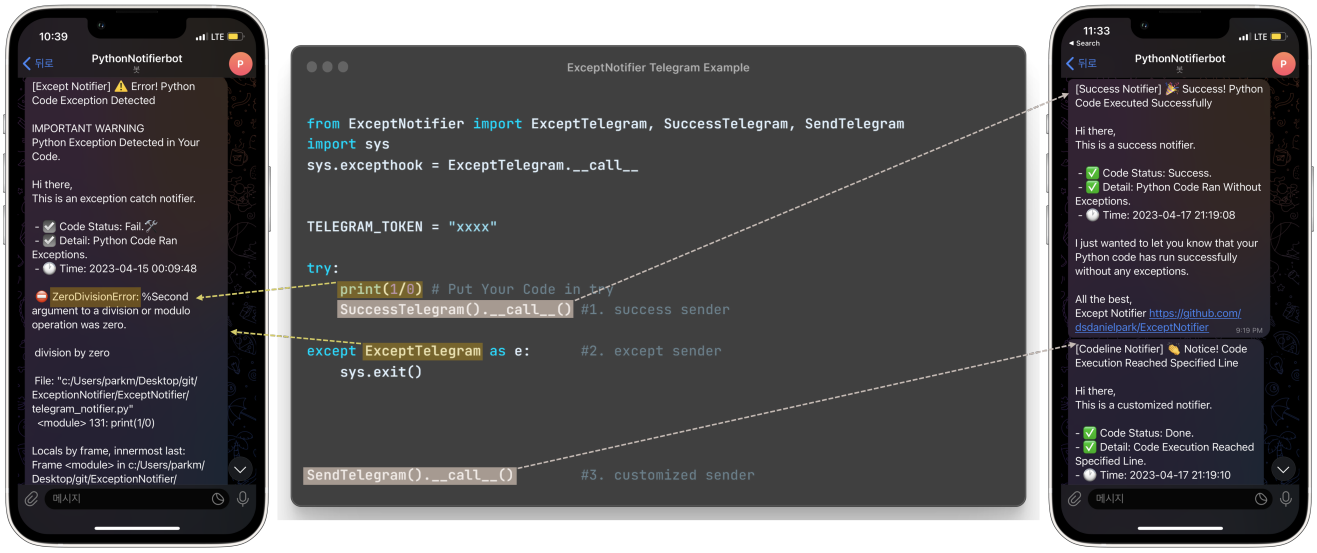
```
from ExceptNotifier import send_telegram_msg

_TELEGRAM_TOKEN =  "xxxxx"

send_telegram_msg(_TELEGRAM_TOKEN, "This is test message")
```

Applied in each application

You can receive debugging information from ChatGPT via OpenAI's API when using the Except statement. The syntax remains the same, but you'll need to configure these two variables: `_OPEN_AI_MODEL`,`_OPEN_AI_API`

Telegram

Open in Colab As all classes function the same, the example will only use one image, like in Telegram.



- a. Open your telegram app and search for BotFather. (A built-in Telegram bot that helps users create custom Telegram bots)
- b. Type /newbot to create a new bot
- c. Give your bot a name & a username
- d. Copy your new Telegram bot's token
- e. You have to click `Start_bot` and must enter anything to your bot.
    - Before use Notifier, Please use this to check if you follow guide. The Telegram bot may have a slight delay and it responded within 2-3 minutes. *API TEST*

```
from ExceptNotifier import send_telegram_msg

_TELEGRAM_TOKEN = "xxxxx:xxxxx-xxxx"
send_telegram_msg(_TELEGRAM_TOKEN, 'msg')
```

For more infomation, visit Telegram Bot Father API

a. Notifier without OpenAI API

Notifier

```
from ExceptNotifier import ExceptTelegram, SuccessTelegram, SendTelegram
import sys, os
sys.excepthook = ExceptTelegram.__call__
os.environ['_TELEGRAM_TOKEN'] = "xxxx"

try:
    print(1/0)
    SuccessTelegram().__call__() #1. success sender

except ExceptTelegram as e:      #2. except sender
    sys.exit()
```

```
SendTelegram().__call__()        #3. customized sender
```

b. Notifier with OpenAI API

- If you just set `_OPEN_AI_API` and `_OPEN_AI_MODEL` environment variables in all application use case, AI MODEL will automatically send debugging information as a message. Currently, it is mainly based on the `GPT-3.5-TURBO` model, but we plan to update it so that other models can be used later. *Notifier*

```
from ExceptNotifier import ExceptTelegram, SuccessTelegram, SendTelegram
import sys, os
sys.excepthook = ExceptTelegram.__call__
os.environ['_TELEGRAM_TOKEN'] = "xxxx"
os.envirion['_OPEN_AI_MODEL']="gpt-3.5-turbo"
os.envirion['_OPEN_AI_API']="sk-xxxxxx"

try:
    print(1/0)
    SuccessTelegram().__call__() #1. success sender

except ExceptTelegram as e:      #2. except sender
    sys.exit()

SendTelegram().__call__()        #3. customized sender
```

Mail

In the except statement, an email is sent along with the error message. Additionally, you can send emails from any desired line.

- a. Log in with the sender's email ID.
- b. Obtain an app password for sending Google Mail at the following link or google document.

API TEST

```
from ExceptNotifier import send_gmail_msg

_GMAIL_SENDER_ADDR = 'xxxx@gmail.com'
_GAMIL_RECIPIENT_ADDR = 'xxxx@gmail.com'
_GMAIL_APP_PASSWORD_OF_SENDER = 'xxxx'
subject_msg = "Test Title"
body_msg = "Test Body"

send_gmail_msg(
    _GMAIL_SENDER_ADDR,
    _GAMIL_RECIPIENT_ADDR,
    _GMAIL_APP_PASSWORD_OF_SENDER,
```

```
    subject_msg,
    body_msg)
```

Notifier

```
import sys, os
from ExceptNotifier import ExceptMail, SuccessMail, SendMail
sys.excepthook = ExceptMail.__call__

# Define the next two variables optionally when using OpenAI's API.
# os.environ['_OPEN_AI_MODEL']="gpt-3.5-turbo"
# os.environ['_OPEN_AI_API']="sk-xxxxxx"
os.environ['_GAMIL_RECIPIENT_ADDR'] = 'xxxxxxx@gmail.com'
os.environ['_GMAIL_SENDER_ADDR'] = 'yyyyyy@gmail.com'
os.environ['_GMAIL_APP_PASSWORD_OF_SENDER'] = 'zzzzzz'

try:
    main()                      # Your Code Here
    SuccessMail().__call__()    # No Exception -> Send Success mail.
except ExceptMail:              # Exception -> Send Fail mail.
    pass

SendMail().__call__()           # When Process Ended -> Any Line mail.
```

▶ See Example...

```
import sys, os
from ExceptNotifier import ExceptMail, SuccessMail, SendMail

# Define the next two variables optionally when using OpenAI's API.
# os.environ['_OPEN_AI_MODEL']="gpt-3.5-turbo"
# os.environ['_OPEN_AI_API']="sk-xxxxxx"
os.environ['_GAMIL_RECIPIENT_ADDR'] = 'xxxxxxx@gmail.com'
os.environ['_GMAIL_SENDER_ADDR'] = 'yyyyyy@gmail.com'
os.environ['_GMAIL_APP_PASSWORD_OF_SENDER'] = 'zzzzzz'

sys.excepthook = ExceptMail.__call__

try:
    # 02.Locate your code
    print(1/0)
    SuccessMail().__call__()   # Success Mail

except ExceptMail as e:        # Exception Mail
    sys.exit()
    print(e)

SendMail().__call__()          # Put Any Line: Sending mail
```

▶ Snippet for Python developers...

```
import sys, os
from ExceptNotifier import ExceptMail, SuccessMail, SendMail
sys.excepthook = ExceptMail.__call__

# Define the next two variables optionally when using OpenAI's API.
# os.environ['_OPEN_AI_MODEL']="gpt-3.5-turbo"
# os.environ['_OPEN_AI_API']="sk-xxxxxx"
os.environ['_GAMIL_RECIPIENT_ADDR'] = 'xxxxxxx@gmail.com'
os.environ['_GMAIL_SENDER_ADDR'] = 'yyyyyy@gmail.com'
os.environ['_GMAIL_APP_PASSWORD_OF_SENDER'] = 'zzzzzz'

try:
    'your code'
    SuccessMail().__call__()
except ExceptMail:
    pass

SendMail().__call__()
```

Discord

- a. Select the channel to receive notifications.
- b. Click `Edit Channel` in the upper right corner of the chat window.
- c. Click `Integrations` - `Webhook` - `New Webhook`.
- d. Then click `Copy Webhook`. *API TEST*

```
from ExceptNotifier import send_discord_msg

send_discord_msg(_DISCORD_WEBHOOK_URL, "Any Test Message")
```

Notifier

```
import sys, os
from ExceptNotifier import ExceptDiscord, SuccessDiscord, SendDiscord
sys.excepthook = ExceptDiscord.__call__

# Define the next two variables optionally when using OpenAI's API.
# os.environ['_OPEN_AI_MODEL']="gpt-3.5-turbo"
# os.environ['_OPEN_AI_API']="sk-xxxxxx"
os.environ['_DISCORD_WEBHOOK_URL'] = "xxxxxxxxxxxxxxxxx"


try:
```

```
    print(1/20)
    SuccessDiscord().__call__() #1 success sender
except ExceptDiscord as e:      #2 except sender
    sys.exit()

SendDiscord().__call__()        #3 customized sender
```

Chime

Open in Colab

- a. Select the Chat room to receive notifications.
- b. Click `Room Setting` in the upper right corner.
- c. Click `Manage Webhook and bot`
- d. Create Add Webhook, set it up, then click `Copy Webhook`. *API TEST*

```
from ExceptNotifier import send_chime_msg

send_chime_msg(_CHIME_WEBHOOK_URL, "Any Test Message")
```

Notifier

```
import sys, os
from ExceptNotifier import SuccessChime, ExceptChime, SendChime
sys.excepthook = ExceptChime.__call__

# Define the next two variables optionally when using OpenAI's API.
# os.environ['_OPEN_AI_MODEL']="gpt-3.5-turbo"
# os.environ['_OPEN_AI_API']="sk-xxxxxx"
os.environ['_CHIME_WEBHOOK_URL'] = "xxxxxxxxxxxxxxxxxx"


try:
    print(1/0)
    SuccessChime().__call__() #1 success sender
except ExceptChime as e:      #2 except sender
    sys.exit()

SendChime().__call__()        #3 customized sender
```

Slack

Open in Colab

- a. visit https://api.slack.com/
- b. `Create an app` - `From scratch` - `Create App`
- c. Add webhook: Click `Incoming Webhooks` - Activate Incomming `On` - Add New Webhook to Workspace
- d. Copy `Webhook URL` *API TEST*

```
from ExceptNotifier import send_slack_msg

send_slack_msg(_SLACK_WEBHOOK_URL, "Any Test Message")
```

Notifier

```
import sys
from ExceptNotifier import ExceptSlack, SuccessSlack, SendSlack
sys.excepthook = ExceptSlack.__call__

# Define the next two variables optionally when using OpenAI's API.
# os.environ['_OPEN_AI_MODEL']="gpt-3.5-turbo"
# os.environ['_OPEN_AI_API']="sk-xxxxxx"
os.environ['_SLACK_WEBHOOK_URL'] =
'https://hooks.slack.com/services/xxxxxxxxxxxxxxxxxxx'

try:
    print(1/0)
    SuccessSlack().__call__() #1 success sender
except ExceptSlack as e:      #2 except sender
    sys.exit()

SendSlack().__call__()        #3 customized sender
```

Line

Open in Colab

- a. Register https://notify-bot.line.me/
- b. Go to mypage https://notify-bot.line.me/my/
- c. Click `Generate Token`, enter Service Name and click `1-on-1 chat with LINE` (anything you like)
- d. Copy Token.

API TEST

```
from ExceptNotifier import send_line_msg

send_line_msg(_LINE_NOTIFY_API_TOKEN:, "Any Test Message")
```

Notifier

```
import sys
from ExceptNotifier import ExceptLine, SuccessLine, SendLine
sys.excepthook = ExceptLine.__call__

# Define the next two variables optionally when using OpenAI's API.
# os.environ['_OPEN_AI_MODEL']="gpt-3.5-turbo"
# os.environ['_OPEN_AI_API']="sk-xxxxxx"
os.environ['_LINE_NOTIFY_API_TOKEN'] = 'xxxxxxxxxxx'

try:
    print(1/20)
    SuccessLine().__call__() #1 success sender
except ExceptLine as e:      #2 except sender
    sys.exit()

SendLine().__call__()        #3 customized sender
```

SMS

- a. Sign up for Twilio. https://www.twilio.com/en-us
- b. Click Console in the upper right corner.
- c. Copy the variables provided in the console.

API TEST

```
from ExceptNotifier import send_sms_msg

_TWILIO_SID = 'xxxx'
_TWILIO_TOKEN = "xxxx"
_SENDER_PHONE_NUMBER = "xxxx"
_RECIPIENT_PHONE_NUMBER = "xxxx"

send_sms_msg(
    _TWILIO_SID,
    _TWILIO_TOKEN,
    _SENDER_PHONE_NUMBER,
    _RECIPIENT_PHONE_NUMBER,
    "Any Test Message")
```

Notifier

```
import sys
from ExceptNotifier import ExceptSMS, SuccessSMS, SendSMS
sys.excepthook = ExceptSMS.__call__

# Define the next two variables optionally when using OpenAI's API.
# os.environ['_OPEN_AI_MODEL']="gpt-3.5-turbo"
```

```
# os.environ['_OPEN_AI_API']="sk-xxxxxx"
os.environ['_TWILIO_SID'] = 'xxxx'
os.environ['_TWILIO_TOKEN'] = 'yyyyyy'
os.environ['_RECIPIENT_PHONE_NUMBER']="+aaaaaa",
os.environ['_SENDER_PHONE_NUMBER']="+bbbbbb",

try:
    print(1/10)
    SuccessSMS().__call__() #1 success sender
except ExceptSMS as e:      #2 except sender
    sys.exit()

SendSMS().__call__()        #3 customized sender
```

Teams

- a. Create the channel that you want to notify.
- b. `App` - `Search: webhook` - `Incoming Webhook` https://teams.microsoft.com/l/app/203a1e2c-26cc-47ca-83ae-be98f960b6b2?source=app-details-dialog
- c. Click `Request Approval`
  After you can use webhook incomming. Proceed to next steps. Microsoft Teams allows limited application access per organization, so it can only be used if the webhook incoming application is available.
- c. Go to the team channel to receive notifications, and click `Connectors` in Settings.
- d. `Connectors` After configuring webhook incoming in Connector, copy the webhook URL.

API TEST

```
from ExceptNotifier import send_teams_msg

_TEAMS_WEBHOOK_URL = 'xxxx'

send_teams_msg(_TEAMS_WEBHOOK_URL, "Any Test Message")
```

Notifier

```
import sys
from ExceptNotifier import ExceptTeams, SuccessTeams, SendTeams
sys.excepthook = ExceptTeams.__call__

# Define the next two variables optionally when using OpenAI's API.
# os.environ['_OPEN_AI_MODEL']="gpt-3.5-turbo"
# os.environ['_OPEN_AI_API']="sk-xxxxxx"
os.environ['_TEAMS_WEBHOOK_URL'] = 'microsoft webhook _TEAMS_WEBHOOK_URL'

try:
    print(1/20)
    SuccessTeams().__call__() #1 success sender
```

```
except ExceptTeams as e:      #2 except sender
    sys.exit()

SendTeams().__call__()        #3 customized sender
```

Kakaotalk

- a. Sign up at the following site: https://developers.kakao.com/
- b. Click `My Application` on the top bar.
- c. Click `Add an application`, set a name, and create it.
- d. Click `Kakao Login` in the left menu, then change the State of `Kakao Login Activation` to ON on the resulting page.
- e. In `My Application > Product Settings > Kakao Login`, be sure to set `Redirect URI` as follows: https://example.com/oauth
- f. In the left Consent Items menu, set `Send message in KakaoTalk` to optional agreement.
- g. Copy the `REST API Key` in `My Application > App Settings > Summary`.
- h. If you have successfully completed all of the above steps, go to the following document and follow the instructions: ./tutorials/kakao_token_generator.ipynb

API TEST

```
from ExceptNotifier import send_kakao_msg

_KAKAO_TOKEN_PATH = 'xxx/xx/xxx.json'

send_kakao_msg(_KAKAO_TOKEN_PATH, msg)
```

Notifier

```
import sys
from ExceptNotifier import ExceptKakao, SuccessKakao, SendKakao
sys.excepthook = ExceptKakao.__call__

# Define the next two variables optionally when using OpenAI's API.
# os.environ['_OPEN_AI_MODEL']="gpt-3.5-turbo"
# os.environ['_OPEN_AI_API']="sk-xxxxxx"
os.environ['_KAKAO_TOKEN_PATH'] = 'xxxx/xxx/xxx.json''

try:
    print(1/0)
    SuccessKakao().__call__() #1 success sender
except ExceptKakao as e:      #2 except sender
    sys.exit()

SendKakao().__call__()        #3 customized sender
```

Wechat

a. Get Webhook URL by visiting here *API TEST*

```
from ExceptNotifier import send_wechat_msg

send_wechat_msg(_WECHAT_WEBHOOK_URL, msg)
```

Notifier

```
import sys
from ExceptNotifier import ExceptWechat, SuccessWechat, SendWechat

# Define the next two variables optionally when using OpenAI's API.
# os.environ['_OPEN_AI_MODEL']="gpt-3.5-turbo"
# os.environ['_OPEN_AI_API']="sk-xxxxxx"
os.environ['_WECHAT_WEBHOOK_URL'] = "xxxxxxxxxxx"
sys.excepthook = ExceptWechat.__call__

try:
    print(1/0)
    SuccessWechat().__call__() #1 success sender
except ExceptWechat as e:      #2 except sender
    sys.exit()

SendWechat().__call__()        #3 customized sender
```

Beep

No setup is required. Use as follows. *TEST*

```
from ExceptNotifier import beep

beep(sec=1, freq=1000)
```

Notifier

```
from Exceptnotifier import ExceptBeep, SuccessBeep, SendBeep(), beep()
os.environ['BEEP_TIME'] = 1
sys.excepthook = ExceptBeep.__call__

try:
    print(1/20)
    SuccessBeep().__call__() #1 success beep-beep
```

```
except ExceptBeep as e:      #2 except beep-beep
    sys.exit()

SendBeep().__call__()        #3 customized beep-beep

beep()
```

Desktop

No setup is required. Use as follows. *TEST*

```
from ExceptNotifier import send_desktop_msg

title_msg = "Test Title"
body_msg = "Test Body"
DISP_TIME = 5

send_desktop_msg(title_msg, body_msg, DISP_TIME)
```

Notifier

```
from ExceptNotifier import ExceptDesktop, SuccessDesktop, SendDesktop
sys.excepthook = ExceptDesktop.__call__
# Define the next two variables optionally when using OpenAI's API.
# os.environ['_OPEN_AI_MODEL']="gpt-3.5-turbo"
# os.environ['_OPEN_AI_API']="sk-xxxxxx"

try:
    print(1/0)
    SuccessDesktop().__call__() #1 success sender

except ExceptDesktop as e:      #2 except sender
    sys.exit()

SendDesktop().__call__()        #3 customized sender
```

Contributing Guide

I will announce contributing rules when the code development status changes to beta soon. Until then, please create an issue for feature requests and bug reports. I would greatly appreciate it if you use it a lot and give your opinions generously. Thank you sincerely.

License

MIT

Code of Conduct

Everyone participating in the `ExceptNotifier` project, and in particular in the issue tracker, pull requests, and social media activity, is expected to treat other people with respect and more generally to follow the guidelines articulated in the Python Community Code of Conduct.

Contacts

Core maintainers: Daniel Park, South Korea
Email parkminwoo1991@gmail.com

- Developer note: Link

Could you kindly add this badge to your repository?

```
![Except-Notifier](https://img.shields.io/badge/pypi-ExceptNotifier-orange)
```

I would appreciate it if you could share the document widely by specifying that the source is the ExceptNotifier official github.

The package is currently in the development and QA stages, and the development stage will be updated at the top of this page. If it is determined that the product is stable through feature improvement, addition, and issue resolution, the development stage will reach stage 5. If no new updates or issues arise, it will be adjusted upward to stage 6 or higher.