

[ChatGPT를 활용한 파이썬코드 작성]

○ Email 발송



한국의 네이버 메일을 이용해서 메일을 발송하는 파이썬 코드를 작성해 줘



네이버 메일을 이용하여 파이썬 코드를 통해 이메일을 발송하는 방법은 네이버 SMTP 서버를 사용하는 것입니다. 아래는 네이버 메일을 이용하여 이메일을 발송하는 예제 코드입니다. 코드 실행 전에 `smtpplib` 라이브러리가 설치되어 있는지 확인해주세요.

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

# 발신자 정보
sender_email = "your_email@naver.com"
sender_password = "your_password"

# 수신자 정보
recipient_email = "recipient_email@example.com"

# 이메일 제목과 내용
subject = "네이버 메일을 통한 이메일 발송"
body = "안녕하세요, 이메일 내용입니다."

# MIME 멀티파트 메시지 생성
message = MIMEMultipart()
message["From"] = sender_email
message["To"] = recipient_email
message["Subject"] = subject

# 메일 내용 추가
message.attach(MIMEText(body, "plain"))

# SMTP 서버 연결
smtp_server = "smtp.naver.com"
smtp_port = 465

try:
    smtp = smtplib.SMTP_SSL(smtp_server, smtp_port)
    smtp.login(sender_email, sender_password)

    # 이메일 발송
    smtp.sendmail(sender_email, recipient_email, message.as_string())
    smtp.quit()

    print("이메일 발송 성공")
except Exception as e:
    print("이메일 발송 실패:", str(e))
```

위 코드에서 ``sender_email``과 ``sender_password``에는 발신자의 네이버 이메일 주소와 비밀번호를 입력하고, ``recipient_email``에는 수신자의 이메일 주소를 입력합니다. 또한 ``subject``와 ``body``에는 이메일의 제목과 내용을 입력합니다.

SMTP 서버에 연결한 후 ``smtp.login()``을 사용하여 발신자의 이메일 주소와 비밀번호로 로그인합니다. 그런 다음 ``smtp.sendmail()``을 사용하여 이메일을 발송합니다. 마지막으로 ``smtp.quit()``을 호출하여 SMTP 서버와의 연결을 종료합니다.

코드 실행 시 발신자 이메일 주소와 비밀번호를 올바르게 입력하고, 네이버 메일 서비스에 접근할 수 있는 인터넷 연결이 필요합니다.

```
(sjcuEnv) c:\_Lecture\SJCU\sjcu-python> c: && cd c:\_Lecture\SJCU\sjcu-python && cmd /C "c:\_Lecture\SJCU\sjcu-python\sjcuEnv\Scripts\python.exe c:\Users\chris\.vscode\extensions\ms-python.python-2023.8.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher 23295 -- c:\_Lecture\SJCU\sjcu-python\14week\autoMail.py "
```

이메일 발송 성공

네이버 메일을 통한 이메일 발송 받은편지함 x



@naver.com

나에게 ▼

안녕하세요, 이메일 내용입니다.

일정 시간 마다 실행, 스케줄러



15초에 1번씩 현재 폴더의 파일 목록을 출력하는 파이썬 코드를 작성해줘



아래는 15초마다 현재 폴더의 파일 목록을 출력하는 파이썬 코드 예시입니다. 이 코드는 `os` 모듈을 사용하여 현재 폴더의 파일을 가져옵니다. `time` 모듈을 사용하여 15초마다 반복되는 루프를 만들어 파일 목록을 출력합니다.

python

Copy code

```
import os
import time

while True:
    file_list = os.listdir()
    print("Current files in the folder:")
    for file in file_list:
        print(file)
    print("-----")
    time.sleep(15)
```

이 코드를 실행하면 현재 폴더의 파일 목록이 15초마다 출력됩니다. `time.sleep(15)` 함수는 15초 동안 프로그램을 일시적으로 멈추어줍니다. 필요에 따라 코드를 수정하여 출력 형식이나 출력 대상을 변경할 수 있습니다.



scheduler 패키지를 이용해서 다시 작성해 줘



아래는 `scheduler` 패키지를 사용하여 15초마다 현재 폴더의 파일 목록을 출력하는 파이썬 코드 예시입니다. 이 코드는 `os` 모듈을 사용하여 현재 폴더의 파일을 가져오고, `scheduler` 패키지를 사용하여 일정 시간마다 작업을 반복 실행합니다.

```
import os
import sched
```

```
import time

def print_file_list():
    file_list = os.listdir()
    print("Current files in the folder:")
    for file in file_list:
        print(file)
    print("-----")

# 스케줄러 객체 생성
scheduler = sched.scheduler(time.time, time.sleep)

# print_file_list 함수를 15초마다 반복 실행하는 작업 예약
scheduler.enter(15, 1, print_file_list, ())

# 스케줄러 실행
scheduler.run()
```

이 코드를 실행하면 `print_file_list` 함수가 15초마다 호출되어 현재 폴더의 파일 목록이 출력됩니다. `scheduler.enter(15, 1, print_file_list, ())` 코드는 `print_file_list` 함수를 15초 후에 실행하도록 예약합니다. `scheduler.run()`은 스케줄러를 실행하여 예약된 작업을 반복 실행합니다.

참고로, `scheduler` 패키지는 파이썬 표준 라이브러리인 `sched` 모듈을 기반으로 한 더 유연한 인터페이스를 제공하는 패키지입니다. `sched` 모듈만 사용하여도 비슷한 결과를 얻을 수 있으나, `scheduler` 패키지는 좀 더 편리한 기능과 구문을 제공합니다.



`sched.scheduler(time.time, time.sleep)`의 의미를 설명해줘



``sched.scheduler(time.time, time.sleep)``는 ``sched`` 모듈의 ``scheduler`` 클래스를 생성하는 부분입니다. ``scheduler`` 클래스는 일정을 관리하고 작업을 예약하고 실행하기 위한 스케줄러 객체를 생성합니다.

여기서 ``time.time``과 ``time.sleep``는 ``scheduler`` 객체에 전달되는 두 개의 함수입니다.

- ``time.time``: 현재 시간을 반환하는 함수입니다. 스케줄러는 이 함수를 사용하여 작업을 예약하는 시간을 결정합니다. 일정 시간이 되면 예약된 작업이 실행됩니다.
- ``time.sleep``: 일정 시간 동안 프로그램 실행을 일시적으로 중지하는 함수입니다. 스케줄러는 이 함수를 사용하여 작업 실행 후 다음 작업을 예약하기 전에 일정 시간 동안 대기합니다.

스케줄러 객체를 생성할 때 ``time.time`` 함수를 첫 번째 인자로, ``time.sleep`` 함수를 두 번째 인자로 전달하는 이유는 스케줄러가 정확한 시간을 추적하고 작업을 예약하고 실행하기 위해서입니다. ``time.time`` 함수는 현재 시간을 기준으로 작업을 예약하고, ``time.sleep`` 함수는 일정 시간 동안 대기하는 역할을 합니다.

즉, ``scheduler`` 객체를 생성할 때 ``time.time``과 ``time.sleep`` 함수를 전달함으로써 스케줄러가 시간 관리와 작업 예약/실행을 올바르게 처리할 수 있도록 도와줍니다.



15초마다 반복하지 않은데, 나는 15초마다 1번씩 반복하는 코드를 원해



알겠습니다. 아래는 ``sched`` 모듈을 사용하여 15초마다 1번씩 반복하는 코드 예시입니다.



os 모듈을 못찾는다고 오류발생



죄송합니다. `os` 모듈을 사용하기 위해서는 `os` 모듈을 임포트해야 합니다. 코드에 `import os` 문을 추가하여 `os` 모듈을 임포트하도록 수정하겠습니다.

다음은 수정된 코드 예시입니다:

```
import os
import sched
import time

def print_file_list():
    file_list = os.listdir()
    print("Current files in the folder:")
    for file in file_list:
        print(file)
    print("-----")

# 스케줄러 객체 생성
scheduler = sched.scheduler(time.time, time.sleep)

# print_file_list 함수를 15초마다 1번씩 반복 실행하는 작업 예약
def repeat_print_file_list():
    print_file_list()
    scheduler.enter(15, 1, repeat_print_file_list, ())

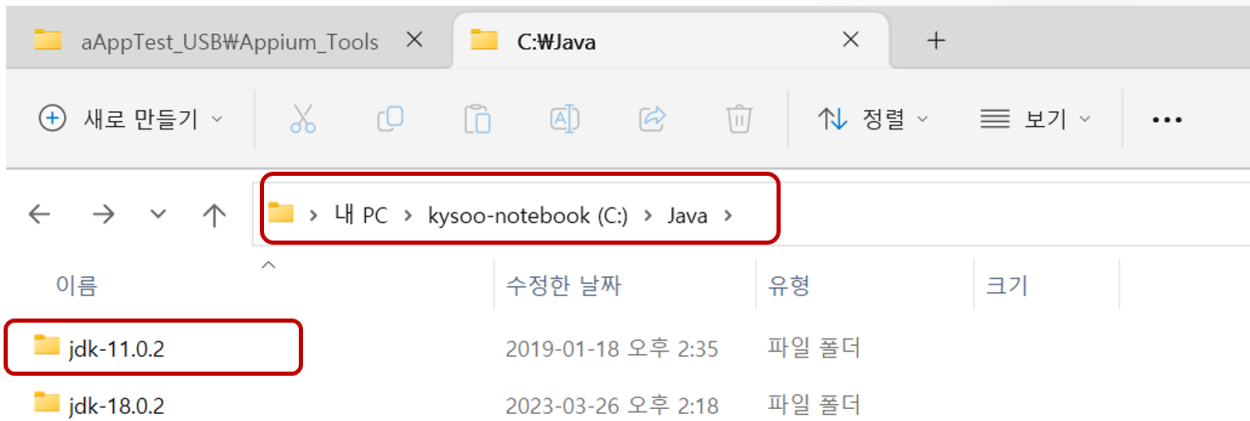
# 첫 번째 작업 예약
scheduler.enter(15, 1, repeat_print_file_list, ())

# 스케줄러 실행
scheduler.run()
```

OpenJDK 설치

프로그램 설치와 동작 확인 - JAVA/JDK (1)

JAVA/JDK를 설치 및 환경 구성 후, 동작을 확인합니다.



- OpenJDK 설치
 - ✓ OpenJDK 압축을 푼 후 대상 폴더로 복사(이동)
 - ✓ 설치 경로를 'C:\Java'로 지정
(JDK폴더 이름은 버전에 따라 결정)
 - ✓ 최종 경로 : C:\Java\jdk-11.0.2

※ 실습 환경을 통일하기 위해 'C:\Java'를 기준 폴더로 선택하였으나, 다른 폴더를 이용해도 괜찮습니다.

프로그램 설치와 동작 확인 - JAVA/JDK (2)

JAVA/JDK를 설치 및 환경 구성 후, 동작을 확인합니다.

[참고]

- 1) Windows + R
- 2) sysdm.cpl, 3 (실패 앞에 빈칸, 실패 뒤에는 빈칸 없음)

The image shows three screenshots of Windows system settings:

- System Properties (시스템 속성):** The 'Environment' (환경) tab is selected. The 'Environment Variables' (환경 변수(N)...) button is highlighted with a red box.
- System Variable Edit (시스템 변수 편집):** The 'Variable name' (변수 이름(N)) is 'JAVA_HOME' and the 'Variable value' (변수 값(V)) is 'C:\Java\jdk-11.0.2'. Both fields are highlighted with a red box. A callout points to this box with the text '시스템 변수에 설정(권장)'.
- Environment Variable Edit (환경 변수 편집):** The 'Path' variable is selected. The value '%JAVA_HOME%\bin' is added to the list and highlighted with a red box. A callout points to this box with the text '시스템 변수에 설정(권장)'. Another callout points to the variable name with the text '%JAVA_HOME%\bin'.

Below the screenshots, the following steps are listed:

- 시스템 변수, 환경변수 설정
 - ✓ JAVA_HOME = JDK 설치 폴더
 - ✓ PATH에 %JAVA_HOME%\bin 추가

프로그램 설치와 동작 확인 - JAVA/JDK (3)

JAVA/JDK를 설치 및 환경 구성 후, 동작을 확인합니다.

```

Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\chris>java --version
openjdk 11.0.2 2019-01-15
OpenJDK Runtime Environment 18.9 (build 11.0.2+9)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.2+9, mixed mode)

C:\Users\chris>
  
```

- JAVA 동작 확인

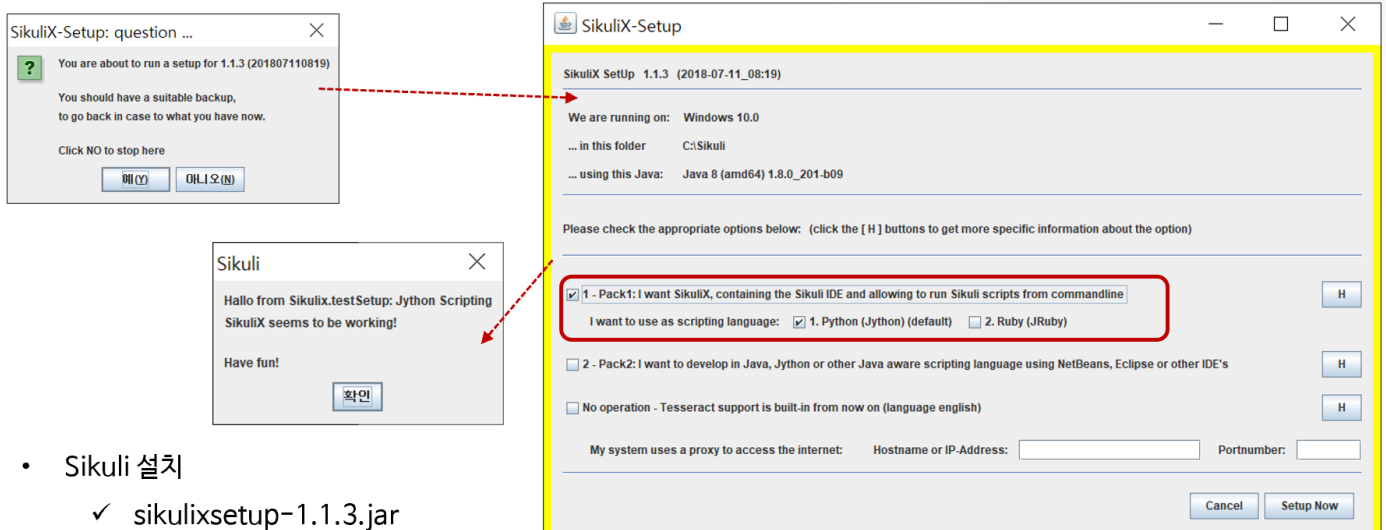
- ✓ java -version
- ✓ javac -version

[참고]

‘-’ 대신 ‘--’를 사용하는 경우도 있습니다.

프로그램 설치와 동작 확인 - Sikuli (1)

Sikuli 프로그램을 설치(다운로드)하고 동작을 확인합니다.



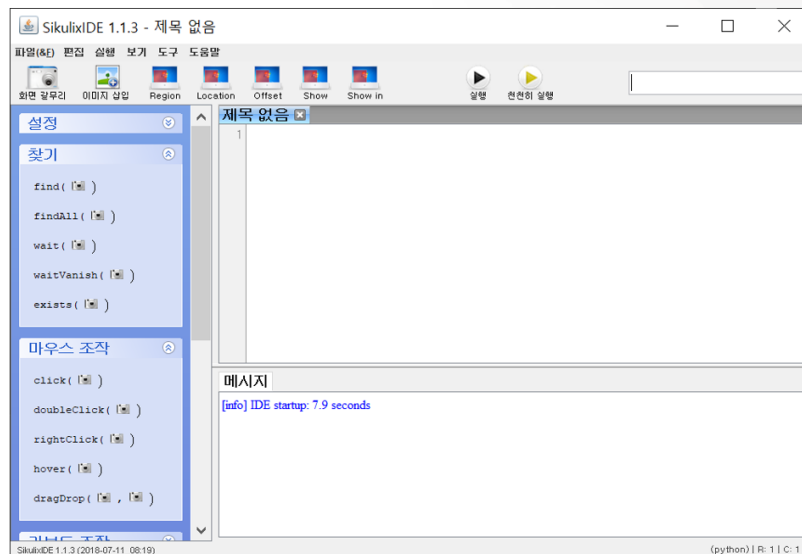
- Sikuli 설치

- ✓ sikulixsetup-1.1.3.jar
- ✓ c:\₩MobileAppTest₩SikuliX 혹은 실습 프로그램 설치 경로를 지정
- ✓ 설치에 필요한 모든 파일은

<https://raiman.github.io/SikuliX1/downloads.html> 에서 다운로드 가능

프로그램 설치와 동작 확인 - Sikuli (2)

Sikuli 프로그램을 설치하고 동작을 확인합니다.



- Sikuli 동작 확인

- ✓ `java -jar sikulix.jar` 혹은 `runsikulix.cmd` (권장) 실행

※ 설치 폴더에 있으며, 관리자 권한으로 실행하는 것을 권장합니다.

※ 경고(Warning)을 피하기 위한 것으로
일반 권한으로 실행하여도 동작에 문제는 없습니다.