

# 32강 스케줄러

특정 시간이나 정해진 간격에 따라 어떤 함수가 자동으로 실행되는 처리

ex) 1마다 한번씩 자동으로 수행되는 기능, 매일 자정에 자동으로 수행되는 기능

apscheduler 가 파이썬 공식문서에서 소개되고 있으므로 권장

```
$ pip install --upgrade apscheduler
```

## #01. 작업준비

### 1) 패키지 참조

```
from apscheduler.schedulers.background import BackgroundScheduler
from apscheduler.triggers.cron import CronTrigger
from apscheduler.jobstores.base import JobLookupError
import datetime as dt
import time
```

### 2) 스케줄에 따라 자동으로 실행될 기능

```
def myjob(name):
    currentTime = dt.datetime.now()
    timeFormat = currentTime.strftime("%Y/%m/%d %H:%M:%S")
    print("[%s] I'm working.... | %s" % (name, timeFormat))
```

```
myjob("lee")
```

```
[lee] I'm working.... | 2023/12/01 11:07:02
```

## #02. 스케줄러 등록

### 1) 정해진 간격마다 실행하기

매 3초마다 실행

스케줄러 객체 생성

```
sched = BackgroundScheduler()
sched.start()
```

스케줄 작업 등록

seconds, minute, hour, day, month, year, week, day\_of\_week, start\_date, end\_date 등을 설정할 수 있다.

start\_date, end\_date는 datetime 객체로 설정해야 함

```
sched.add_job(myjob, "interval", seconds=3, args=["kim"], id='myjob1')
```

```
<Job (id=myjob1 name=myjob)>
```

```
[kim] I'm working.... | 2023/12/01 11:09:17
[kim] I'm working.... | 2023/12/01 11:09:20
[kim] I'm working.... | 2023/12/01 11:09:23
[kim] I'm working.... | 2023/12/01 11:09:26
```

## 스케줄러에서 작업 제거

```
sched.remove_job("myjob1")
sched.shutdown()
```

## 2) cron 표현식으로 설정하기

### cron 표현식

Linux, Mac 등에서 작업 스케줄러를 등록할 때 사용하는 시간 단위 설정 표현식

공백으로 구분하는 7자리의 값으로 구성됨

```
* * * * *
```

각 자리는 순서대로 <초> <분> <시> <일> <월> <요일> <년> 을 의미함

### 값의 설정 방법

필드	허용되는 값	허용되는 특수문자
초 (Seconds)	0~59	, - * /
분 (Minutes)	0~59	, - * /
시 (Hours)	0~23	, - * /
일 (Day of month)	1~31	, - * / L W
월 (Month)	1~12 또는 JAN ~ DEC	, - * /
요일 (Day of week)	0 ~ 6 또는 SUN ~ SAT	, - * / L #
년 (Year)	1970 ~ 2099	, - * /

### 특수문자의 의미

- \* : 모든 값을 뜻합니다.
- ? : 특정한 값이 없음을 뜻합니다.
- : 범위를 뜻합니다. (예) 월요일에서 수요일까지는 MON-WED로 표현
- , : 특별한 값일 때만 동작 (예) 월,수,금 MON,WED,FRI

- \* **/** : 시작시간 / 단위 (예) 0분부터 매 5분 0/5
- \* **L** : 일에서 사용하면 마지막 일, 요일에서는 마지막 요일(토요일)
- \* **W** : 가장 가까운 평일 (예) 15W는 15일에서 가장 가까운 평일 (월 ~ 금)을 찾음
- \* **#** : 몇째주의 무슨 요일을 표현 (예) 3#2 : 2번째주 수요일

CRON 표현식	빈도
0 0/5 * * * ?	5분마다
0 0 12 * * ?	매일 낮 12시에
0 15 10 ? * *	매일 오전 10:15에
0 15 10 * * ?	매일 오전 10:15에
0 15 10 * * ? *	매일 오전 10:15에
0 15 10 * * ? 2014	2014년 동안 매일 오전 10:15에
0 * 14 * * ?	매일 오후 2:00에 시작해서 매 분마다 실행하고 오후 2:59에 끝남
0 0/5 14 * * ?	매일 오후 2:00에 시작해서 5분마다 실행하고 오후 2:55에 끝남
0 0/5 14,18 * * ?	매일 오후 2시에 시작해서 5분마다 실행되어 오후 2:55에 끝나고, 오후 6시에 시작해서 5분마다 실행되어 오후 6:55에 끝남
0 0-5 14 * * ?	매일 오후 2:00에 시작해서 매 분마다 실행하고 오후 2:05에 끝남
0 10,44 14 ? 3 WED	3월 동안 매주 수요일 오후 2:10과 오후 2:44에
0 15 10 ? * MON-FRI	주중 오전 10:15에
0 15 10 15 * ?	매달 15일 오전 10:15에
0 15 10 L * ?	매월 말일 오전 10:15에
0 15 10 ? * 6L	매월 마지막 금요일 오전 10:15에
0 15 10 ? * 6L	매월 마지막 금요일 오전 10:15에
0 15 10 ? * 6L 2014-2017	2014년부터 2017년까지 매월 마지막 금요일 오전 10:15에
0 15 10 ? * 6 3	매월 세 번째 금요일 오전 10:15에
0 0 12 1/5 * ?	해당 월의 첫날부터 시작하여 매월 5일 마다 평일 정오에
0 11 11 11 11 ?	11월 11일 오전 11:11마다

```
sched = BackgroundScheduler()
sched.start()
```

```
# 2초마다가 아닌, 매 분 2초가 됨
# -> 2 * * * * *
sched.add_job(myjob, 'cron', second=2, args=["kim"], id='myjob2')

# 매초(*)마다 2초 간격(/2)으로~
sched.add_job(myjob, 'cron', second='*/2', args=["hong"], id='myjob3')
```

```
<Job (id=myjob3 name=myjob)>
```

```
[hong] I'm working.... | 2023/12/01 11:38:24
[hong] I'm working.... | 2023/12/01 11:38:26
[hong] I'm working.... | 2023/12/01 11:38:28
[hong] I'm working.... | 2023/12/01 11:38:30
[hong] I'm working.... | 2023/12/01 11:38:32
[hong] I'm working.... | 2023/12/01 11:38:34
[hong] I'm working.... | 2023/12/01 11:38:36
[hong] I'm working.... | 2023/12/01 11:38:38
[hong] I'm working.... | 2023/12/01 11:38:40
[hong] I'm working.... | 2023/12/01 11:38:42
[hong] I'm working.... | 2023/12/01 11:38:44
[hong] I'm working.... | 2023/12/01 11:38:46
[hong] I'm working.... | 2023/12/01 11:38:48
[hong] I'm working.... | 2023/12/01 11:38:50
[hong] I'm working.... | 2023/12/01 11:38:52
[hong] I'm working.... | 2023/12/01 11:38:54
[hong] I'm working.... | 2023/12/01 11:38:56
[hong] I'm working.... | 2023/12/01 11:38:58
[hong] I'm working.... | 2023/12/01 11:39:00
[kim] I'm working.... | 2023/12/01 11:39:02[hong] I'm working.... | 2023/12/01 11:39:04
[hong] I'm working.... | 2023/12/01 11:39:06
[hong] I'm working.... | 2023/12/01 11:39:08
```

```
sched.remove_job("myjob2")
sched.remove_job("myjob3")
sched.shutdown()
```

### 3) 특정 시각에 수행

```
sched = BackgroundScheduler()
sched.start()
```

```
targetDate = dt.datetime(2023, 12, 1, 11, 43, 0)
```

```
sched.add_job(myjob, 'date', run_date=targetDate, args=["park"], id='myjob4')
```

```
<Job (id=myjob4 name=myjob)>
```

```
[park] I'm working.... | 2023/12/01 11:43:00
```

```
# 예약된 시각에 1회 수행하고 종료하므로 remove_job을 처리할 필요가 없다.  
# sched.remove_job("myjob4")  
sched.shutdown()
```

### #03. 메일링 리스트 개선

```
# 앞 단원에서 구현한 메일 발송 모듈
```

```
import MyMailer
```

```
# 비동기 처리 기능을 제공하는 모듈
```

```
import concurrent.futures as futures
```

```
today = dt.datetime.now()  
year = today.year  
month = today.month  
day = today.day
```

```
fromAddr = "운영지원팀 <leekh4232@gmail.com>"  
subjectTpl = "{name}님의 {yy}년 {mm}월 급여명세서입니다."
```

```
with open('mail/content.txt', 'r', encoding='utf-8') as f:  
    contentTpl = f.read()
```

```
def sendmail():
```

```
    startTime = dt.datetime.now()
```

```
    with open("mail/mail_list.csv", "r", encoding="euc-kr") as f:  
        csv = f.readlines()
```

```
    with futures.ThreadPoolExecutor(max_workers=10) as executor:
```

```
        for line in csv:
```

```
            name, email, file1, file2 = line.strip().split(",")
```

```
            toAddr = "{name} <{email}>".format(name=name, email=email)
```

```
            subject = subjectTpl.format(name=name, yy=year, mm=month)
```

```
            content = contentTpl.format(name=name, yy=year, mm=month, dd=day)
```

```
            #MyMailer.sendMail(fromAddr, toAddr, subject, content, [file1, file2])
```

```
            executor.submit(MyMailer.sendMail, fromAddr, toAddr, subject, content,
```

```
    endTime = dt.datetime.now()
```

```
    workTime = endTime - startTime
```


```
    print("작업에 소요된 시간은 총 %s초 입니다." % workTime.seconds)
```

```
sched = BackgroundScheduler()
sched.start()

sched.add_job(sendmail, 'cron', second='*/5', id='mymailer')
```


<Job (id=mymailer name=sendmail)>

Execution of job "sendmail (trigger: cron[second='\*/5'], next run at: 2023-12-01 11:54



작업에 소요된 시간은 총 5초 입니다.  
작업에 소요된 시간은 총 4초 입니다.  
작업에 소요된 시간은 총 4초 입니다.

Execution of job "sendmail (trigger: cron[second='\*/5'], next run at: 2023-12-01 11:54



작업에 소요된 시간은 총 5초 입니다.  
작업에 소요된 시간은 총 4초 입니다.

Execution of job "sendmail (trigger: cron[second='\*/5'], next run at: 2023-12-01 11:54



작업에 소요된 시간은 총 5초 입니다.

```
sched.remove_job("mymailer")
sched.shutdown()
```

작업에 소요된 시간은 총 5초 입니다.